Master Thesis:

A generic model of project management with Vensim

By

Suinan Li

Agder University

Faculty of Engineering and Science

Grimstad

26 May 2008

Abstract

Large projects without good management will often cause cost and deadline overruns, missing the project scope and insufficient quality. The construction of the channel tunnel which link between England and France was budgeted at $7 billion, but it entered service in the second half of 1994 with a price tag of $13 billion. The Disneyland in Paris initially planned to cost $2.25 billion in the project, but finally cost $4 billion. Some projects are even never finished, typically software projects

System dynamics has proven to be an effective methodology to explain the reasons of project failure and to provide insights for best practice in project management. In more than 100 projects where system dynamics has been applied, the cost-benefit ratio is 1:200. More than 50 litigation cases which have used SD methodology have all won their cases in court [1]. Consequently, in this thesis I will implement a generic system dynamics stock-and-flow model with the popular system dynamics modeling tool—Vensim. The causal-loop diagram I decide to implement is represented in the paper "System dynamics applied to project management: a survey, assessment and directions for future research" by Lyneis and Ford (2007). I choose it because it concludes most dynamic factors during the process in one single project. I expect that my generic system dynamics model could be useful in improving project management.

Preface

This thesis is the partial fulfillment of the two-year Master of Science program in Information and Communication Technology (ICT) at University of Agder, Faculty of Engineering and Science in Grimstad, Norway. The thesis has been carried out from January to June 2008 and the workload equals 30 ECTS.

First and foremost, I would like to thank my supervisor Professor Jose.J.Gonzalez at University of Agder. He has been highly supportive throughout the whole project period. I would also like to thank Jing Li who has given me many suggestions during the thesis, and Thomas Jakobsen who is always available for me whenever I need help. Last but not least, I would like to thank Mr. Stein Bergsmark, Mr. Ole-Christoffer Granmo and Mrs. Sissel Andreassen for their coordination of our studies and daily life in Grimstad.

Contents

Figure Content

1. Introduction

1.1. Problem description

Cost and deadline overruns, missing the project scope and insufficient quality typically happen in projects from all kinds of industries. The construction of the channel tunnel which link between England and France was budgeted at $7 billion, but it entered service in the second half of 1994 with a price tag of $13 billion, described in ref. [2]; the Disneyland in Paris initially planned to cost $2.25 billion in the project, but finally cost $4 billion; and the famous case between Ingalls Shipbuilding and the U.S. Navy described in ref. [3]. Some projects are even never finished, typically software projects [4].

That is because firstly, projects, especially large projects, are difficult to plan and manage. Project management has to follow a series of activities of deciding the whole project scope and certain requirements, defining the project schedule and utilizing the limit budget and work force resources well. Secondly, when problems happen during the project, the action that project managers often choose to control the project progress is working overtime. However, this action can result various dynamic feedback responses such as ripple and knock-on effects which actually lead the project progress even worse. This project-overrun situation makes the application of system dynamics to project management a fertile and productive field of study.

In this master thesis, I am going to build a generic system dynamic simulation model according to the causal loop diagrams that are represented in ref. [5]. This model is applied to project management, including all the factors and relationships which are involved and influenced in the progress of the project. Since it is a very complicated dynamic system, I choose to approach the model step by step and provide incremental testing for each step to ensure the "so-far" model is correct.

Although projects from different industries have their own particular characteristics, some similarities are still existed. For example, oil and gas industry has quite different working objects and procedures from software development industry, but they probably both will choose working overtime when the progress is behind the schedule. Therefore, my purpose of building this generic system dynamic model is to provide a general view to project managers of what is going to happen in the process of a complex project and how could they manage to control the projects. Furthermore, I hope my work can provide insights of "Dynamic Stories" through analyzing feedback loops in the model, so that an in-depth study of typical project management failures can be widely conducted.

1.2. Research hypotheses

"Generic" means general, so in order to build this generic model I hypothesize a very common case for my model. I assume that the total work scope of this project is 1200 tasks, two employees are working in the project to begin with and the project is planned to complete in 1200 days. I will also make some other assumptions such as different multipliers in the process of building model.

1.3. Thesis structure

In my thesis, I will review the ref. [5] paper first to give some understanding of project management and explain the causal loop diagrams. Then a methodology will be stated to show how I suppose to do in this project. Afterwards, it is the main part of the thesis—model approach. I will discuss the very details about how I build this complex model incrementally. Finally, some policies analysis and recommendations will be provided.

2. Literature Review

System dynamics is an approach to present and analyze the behavior of a complex system in order to have a well understanding of what is exactly going on within the process [6]. It deals with the feedback loops and time delays that affect the whole system. One of the most successful areas for the application of system dynamics has been project management, because system dynamics has proven to be an effective methodology to explain the reasons of project failure and to provide insights for best practice in project management. Many different types of models have been developed to improve project management, but most of those models are beyond the scope of a single article, such as the computational models developed in ref. [7]. In ref. [5] paper, they focus on models of single projects built using the system dynamics methodology. In the paper, they described model structures into four groups which are *Project Features*, *Rework Cycle*, *Project Control* and *Ripple and Knock-on Effects* and give the related causal loop diagrams according to those groups.

1. Project features: System dynamics focuses on modeling features found in real system. In project these features include development processes, resources, managerial mental models, and decision making. Adding more important components of actual project increases the ability to simulate realistic project dynamics.

2. Rework Cycle: Rework cycle is the canonical structure and the most important feature of system dynamics project model. Cooper has written a series paper to discuss rework cycle in ref. [8] and in those paper he developed the first full rework cycle model, shown conceptually in Figure 1.

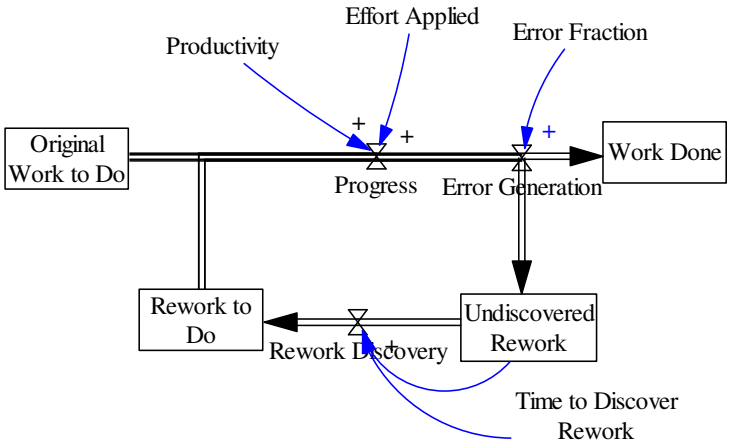


Figure 1: Rework Cycle

There are four stocks in rework cycle. At the start of project, all work stores in the stock *Original Work to Do*. It's easy to imagine that not all works can be finished successfully at once in real life project. An *Error Fraction* measures that could vary in the range of 0 to 1

diverts more or less of the work being done into the rework cycle. Work done correctly enters the *Work Done* stock and never needs rework. However, work which contains errors goes to the stock *Undiscovered Rework*. Errors cannot be immediately detected and this process may take months or even years. Once discovered, the *Rework to Do* backlog demands the application of extra effort. So long as *Error Fraction* is less than 1, some work being done--even rework itself will move into the rework cycle again.

3. Project control: Management actions taken to control a project's performance are modeled as efforts to close the gap between target and actual performance. There are two basic methods available: increasing work intensity even working overtime, or slipping the deadline. Three common actions can be taken when project managers anticipate that they will miss a deadline: hire additional workforce, work overtime and work faster. These actions form the feedback loop of"Add People", "Work More", and "Work Faster/Slack Off " illustrated in Figure 2. In these loops, there is an expected completion delay shown in the lower right of the figure. It indicates more time required to finish the project than the time remaining during the project.

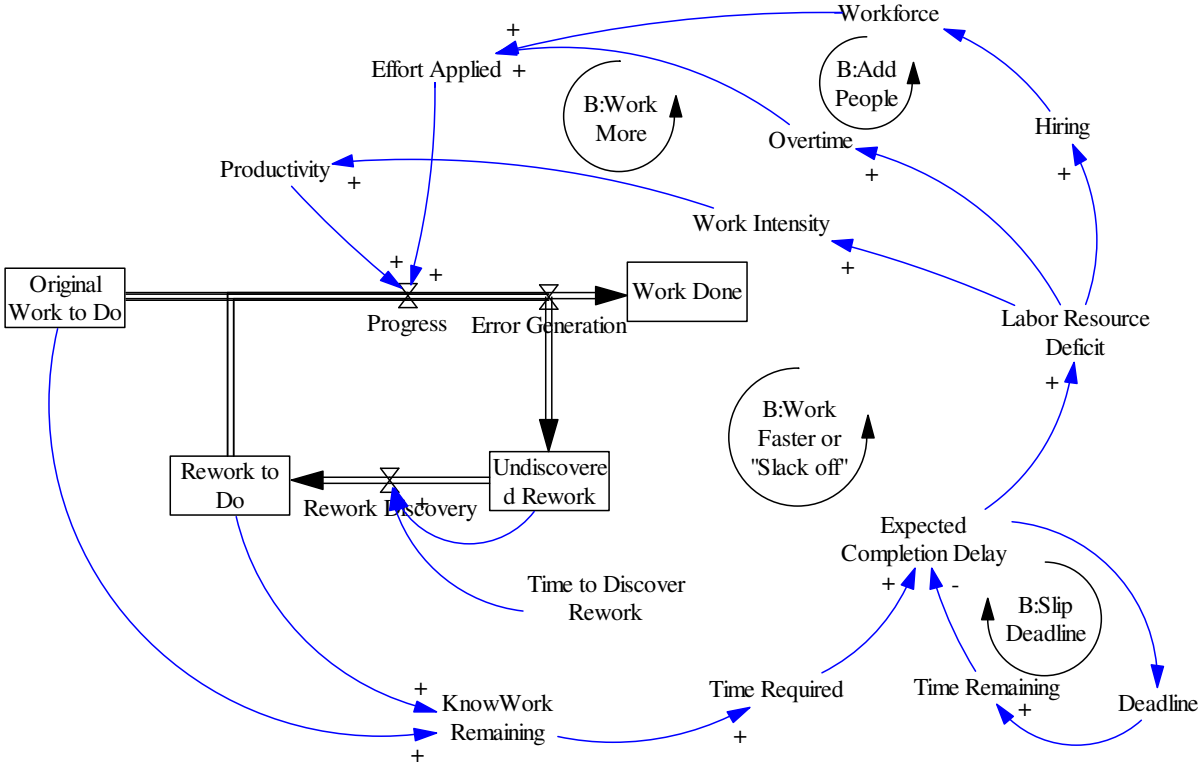


Figure 2: Controlling feedback loops for achieving a target schedule (deadline).

In the figure, the + and – signs at the arrow heads denote polarity. A causal link from A to B is positive if A adds to B, or if a change in A produces a change in B in the same direction. A causal link from A to B is negative if A subtracts from B, or if a change in A produces a change in B in the opposite direction [9].

4. Ripple effects: Actions taken to close a gap between project performance and targets have unintended side effects that generate policy resistance. This kind of effects is commonly

called “ripple effects”. These ripple effects are the primary impacts of project control on rework and productivity. Figure 3 adds four important ripple effect feedbacks of the three project control actions shown in Figure 2.

Hiring can dilute the level of workers’ experience. New employees have less skill and familiarity with the project than experienced ones. Therefore, the new people are less productive and tend to make mistakes. In addition, new employees also require experienced developers to spend time to train them instead of doing development. Larger team size can increase congestion and communication difficulties between team members, which can increase errors production and decrease productivity. Overtime leads to fatigue that also increases errors and decreases productivity. Higher work intensity increases errors, because people just want to finish the job quickly rather than perfectly. Reduced productivity and increased rework keeps the amount of work remaining greater than expected. These effects form the Experience Dilution, Too Big to Manage, Burnout, and Haste Makes Waste loops.

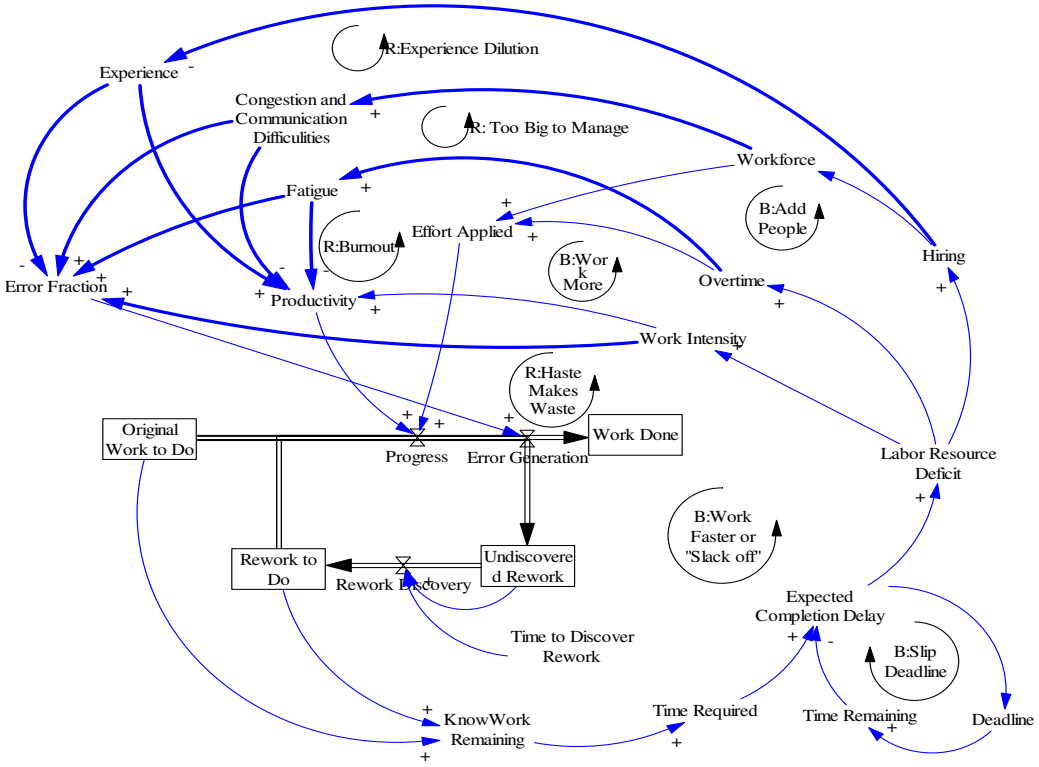


Figure 3: Policy resistances via ripple effects of rework and controlling feedback

5. Knock-on effects: “Knock-on effects” refers to the secondary impacts of project control efforts. They are often the feedback impacts of ripple effects. Some of the knock-on effects are consequences of physical processes related to work flow through projects that transmit from upstream work to downstream work, while others are due to “human” reactions to project conditions.

Figure 4 builds from Figure 3 to illustrate these "knock-on" effects generate significant harmful dynamics in the project. They include:

- "Haste creates out-of-sequence work" — Work should be done step by step following to a physical process. Doing tasks in parallel reduces productivity and increases errors.
- "Errors build errors" —Undiscovered rework very important in the process of project. Undiscovered errors in upstream work products errors in downstream project phases, and then reduce the quality of downstream work. Coded software is a good example of this contamination effect.
- "Errors create more work"—the process of correcting errors can increase the number of tasks that need to be done in order to fix the problem. Taylor and Ford demonstrate a thought of "tipping point" in this feedback loop , at which the increasing of the project scope can stop and start to decline [10]. This often results in project cancellation.
- "Hopelessness"—fatigue and rework can create a sense of "hopelessness". This morale problem would increases errors and reduces productivity, and which also increases turnover.

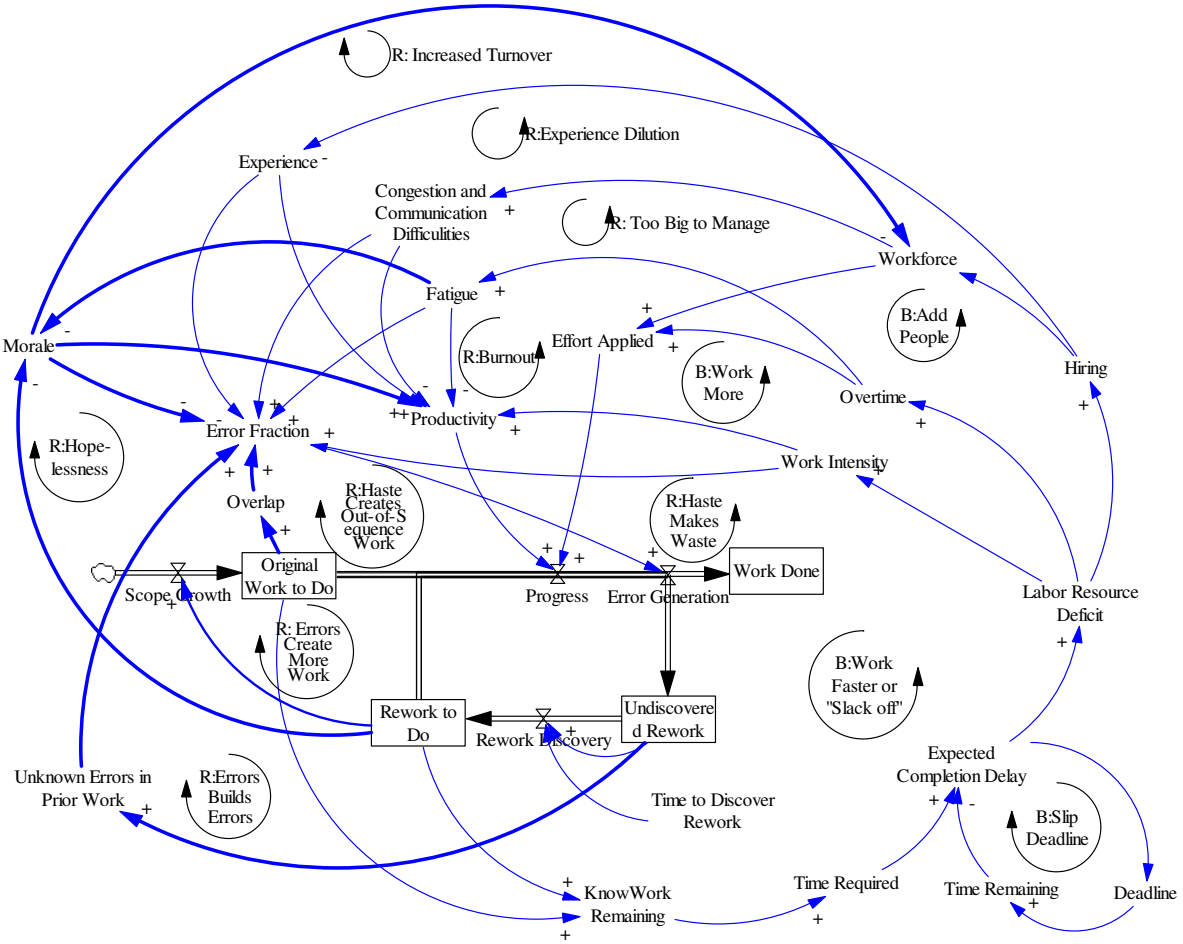


Figure 4: Policy resistances via "knock-on" effects to controlling feedback to improve schedule performance

In this project, my work is to create a generic computer simulation model of project management according to the causal loop diagrams shown above, to simulate it and see how we manage to control the project through the system dynamic model.

3. Methodology

3.1. Methodology of system dynamics

System dynamics is a method for studying the world around us. Unlike other scientists, who study the world by breaking it up into smaller and smaller pieces, system dynamicity look at things as a whole [11]. Many of these systems and problems can be built as models on a computer. System dynamics takes advantage of the fact that a computer model can be of much greater complexity and carry out more simultaneous calculations than the mental model in the human mind. In ref. [11], it also lists six steps to solve a problem with system dynamics:

- ✓ Identifies a problem
- ✓ Develops a dynamic hypothesis explaining the cause of the problem
- ✓ Builds a computer simulation model of the system at the root of the problem
- ✓ Tests the model to be certain that it reproduces the behavior seen in the real world
- ✓ Devises and tests in the model alternative policies that alleviate the problem
- ✓ Implements this solution

3.2. Methodology of the project

In my project, since the causal loop diagrams have been given in Figure 2, Figure 3 and Figure 4, I decide to create the model step by step following these figures. The model was created using the feedback approach of System Dynamics, and implemented in the most used system dynamics modeling tool —Vensim DSS software.

In Figure 2, it illustrates the actions taken to control a project's performance to close the gap between target and actual performance. I take two steps to build Figure 2. Firstly, I create a model without considering working intensity and working overtime. I call this model "the basic model". It shows basic features of the project. Secondly, add the working intensity and working overtime part in the basic model to finish Figure 2. Next, build the ripple effects in the model according to Figure 3. And finally, add knock-on effects base on Figure 4. While building model, I use some previous models as reference sources, such as Abdel-Hamid and Madnick model that represented in ref. [12] and Taylor and Ford's model that are given in ref. [10].

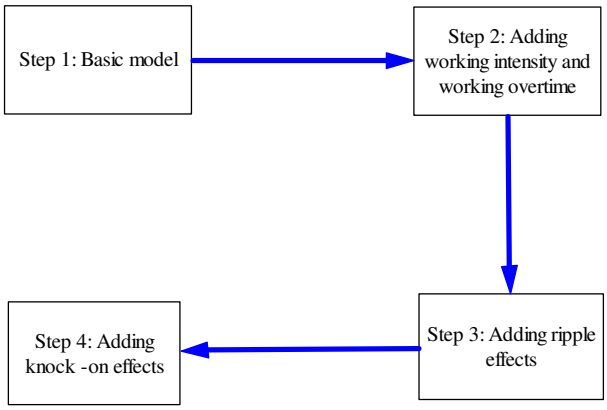


Figure 5: Steps of creating models

Because it is a very complex model, the incremental testing is very important. A problem in the previous step would remain in latter steps and even cause more errors. Therefore, incremental testing has to be done carefully in every step in order to ensure the process of the project is correct so far. I would use model validation in every step to achieve this goal.

Since it is a generic model, there is no specific story about it. During the process of creating model, I have to make a few assumptions for some variables. I will give the explanations of those values in the related parts.

4. Model approach

In this chapter, I describe the very details about how I approach the model incrementally. To begin with, I make some basic assumptions for the project. I assume that at the beginning only two people work on the project; there are 1200 original tasks that need to be done; and the company wants to finish the project 1200 days after starting.

4.1. Step 1: Basic model

Figure 6 shows the structure of basic model.

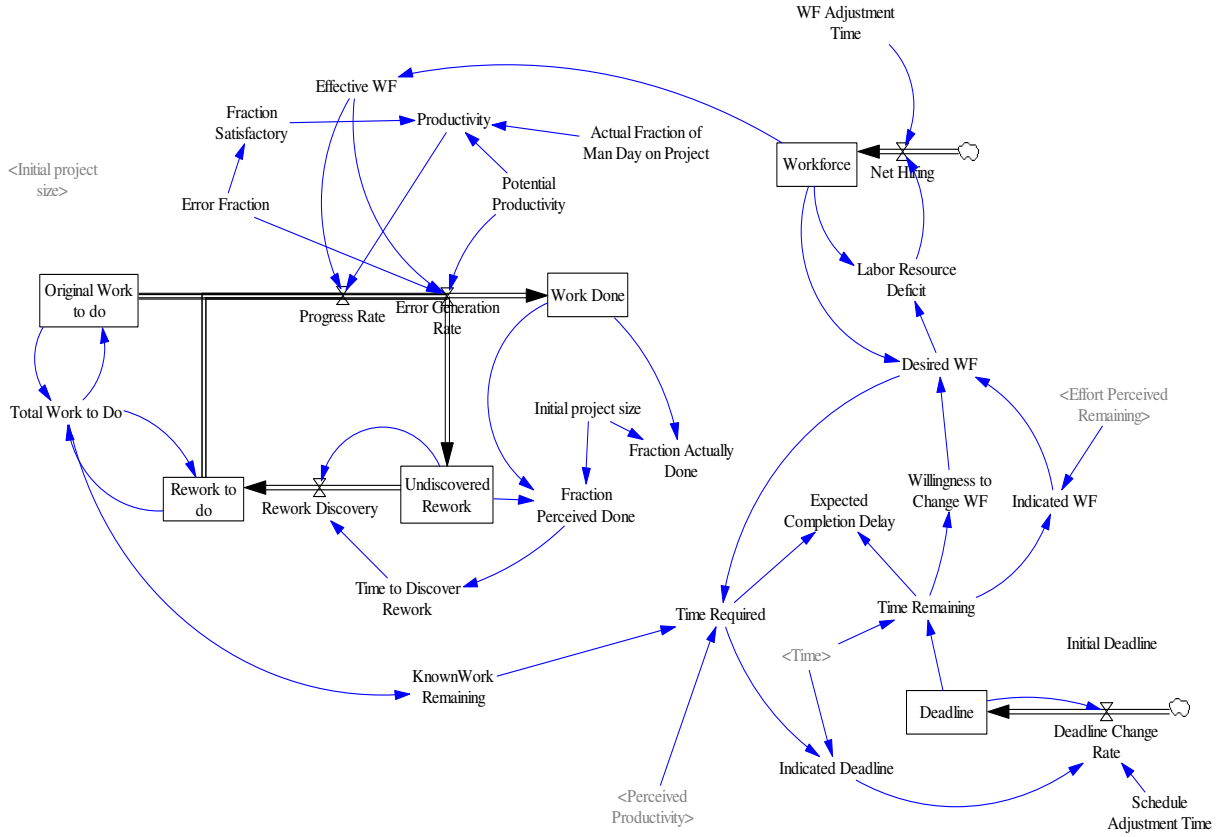


Figure 6: Basic model

4.1.1. Rework Cycle

In this case, there are 1200 original tasks to do at the beginning, so the variable *Original Work to do* is represented as a stock. As the work goes, some of the tasks are done successfully at once and some of them get errors while being done. The two processes here are described by the rates *Progress Rate* and *Error Generation Rate*. Tasks successfully done go to the stock *Work Done*, and unsuccessfully done tasks flow to the stock *Undiscovered Rework*. It takes times to discover rework. Once rework is discovered, the rework backlog required extra effort

to correct it. It is still possible to make errors while rework is being done.

Notice that in the rework cycle there are actually two different progress rates and two different generation rates. They are progress and error generation rate while *Original Work to do* is being done, and, progress and error generation rate while *Rework to do* is being done. But in the model, in order to fit the generic causal-loop model in ref. [5] as much as possible, I just use one *Progress Rate* and one *Error Generation Rate* in the model. These *Progress Rate* and *Error Generation Rate* represent the sum of two progress rates and the sum of error generation rates. Therefore, a problem about how to divide those two different rates comes out. There are many methods to solve this problem in real life. The way I choose here, which might be chosen by project managers too, is using a proportion to solve the problem. In order to do that, I add a variable named *Total Work to Do*, which stands for the total of *Original Work to do* and *Rework to do*. So the *Progress Rate* and *Error Generation Rate* correspond to the *Total Work to do*. By using this variable, I can calculate the proportion between original work and rework in the total work. And I assume that this proportion is the same as the proportion in *Progress Rate* and *Error Generation Rate*, which are shared by original work and rework. So $(Error\ Generation\ Rate)_{original\ work}$= *Error Generation Rate*Original Work to do/Total Work to Do,* $(Progress\ Rate)_{original\ work}$= *Progress Rate*Original Work to do/Total Work to Do*. Similarly, $(Error\ Generation\ Rate)_{rework}$= *Error Generation Rate*Rework to do/Total Work to Do,* $(Progress\ Rate)_{rework}$= *Progress Rate*Rework to do/Total Work to Do.* Accordingly, final related equations are as follows:

Total Work to Do= Original Work to do + Rework to do

*Original Work to do= -Error Generation Rate*Original Work to do/Total Work to Do-Progress Rate*Original Work to do/Total Work to Do. The initial value is Initial project size*

*Rework to do= Rework Discovery-Error Generation Rate*Rework to do/Total Work to Do-Progress Rate*Rework to do/Total Work to Do. The initial value is 0*

4.1.2. Project Schedule

Schedule is one of the important measurements of project performance. Completing projects in time is the goal for every project manager. In my model, I set nominal effort fraction, which is *Actual Fraction of Man Day on Project* in this basic model, to 0.6 rather than 1 because of the waste on slack time. In addition, the quality of work is not 100% – I assume rather only 70 percent of the tasks are done satisfactorily, the remaining 30 percent having errors that are not discovered until later, so the value of *Fraction Satisfactory* is 0.7. Hence, the real *Productivity* in this case, which equals *Fraction Satisfactory*Potential Productivity*Actual Fraction of Man Day on Project*, is less than *Potential Productivity.* This

result would lead the real progress rate lower than the expected value, furthermore affect known work remaining and time required more than expected, and finally the value of *Indicated Deadline* would be bigger than *Deadline*. In order to close the performance gap, I have to move targets toward project behavior, that is, slip the deadline. I assume the company wants to finish the project 1200 days after starting, so the value of initial deadline here is 1200. This value would be increased by a *Deadline Change Rate*, and the schedule adjustment time is set to 180 days.

4.1.3. Workforce

To begin with, two people work on the project, but the company will hire new developers as required. The value of *Total Workforce* would increase by the inflow *Net Hiring* as the project goes. The workforce adjustment time is 90 days. The *Net Hiring* is determined by the difference between total workforce and desired workforce, which is called *Labor Resource Deficit* in the model. Moreover, indicated workforce and the willingness to change workforce are used to define desired workforce. The willingness to hire new staff will decrease along with how little time is left to finish the project. When project approaches to completion, companies have less and less desire to hire more people because it takes much time before newly employees to fully get into the project.

4.1.4. Model validation

I make the model validation by using behavior tests. I compare some simulated results with some common sense expectation to see if these behaviors make sense. Obviously, the main point is to make the model generate the right output for the right reason.

In the basic model, I set the value 0.6 for *Actual Fraction of Man Day on Project*. That means if the actual working hours is 8 per day, people only use 8*0.6=4.8 hours to do the real work, the rest is the slack time. Similarly, we are human beings; people cannot finish every single task successfully at the first time. Errors happen. So in my model I give the value 0.7 to *Satisfactory Fraction*. That is, if 100 tasks have been finished in one day, just 70 of them are really done. Those two values are reasonable in the real life and that is why most projects cannot be accomplished before the desired deadline. Accordingly, I set the model in an ideal condition, which means both values of *Actual Fraction of Man Day on Project* and *Satisfactory Fraction* are 1; the project should be accomplished on the desire deadline which is 1200 days in this case. So I put 1 in both *Actual Fraction of Man Day on Project* and *Satisfactory Fraction* in the basic model to test it. The result graphs shows in Figure 8.

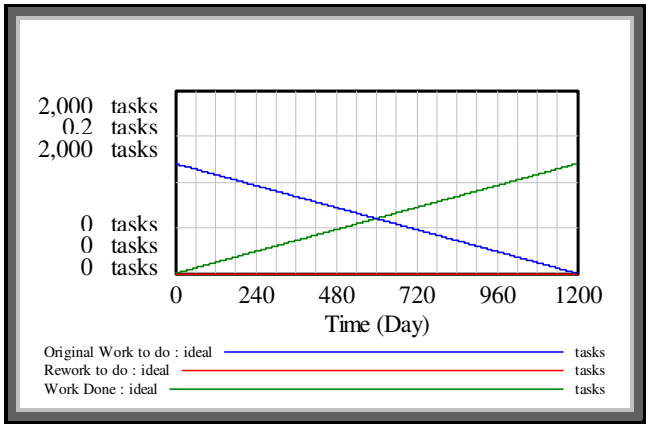


Figure 7: Project Progress in ideal condition

In Figure 7, we can see that the original works are finished exactly on the day 1200. This behavior matches the analysis that I gave above. We also can see that *Rework to do* is 0 all the time in the figure. Since *Satisfactory Fraction* is always 1, no rework need to do, this behavior makes sense too.

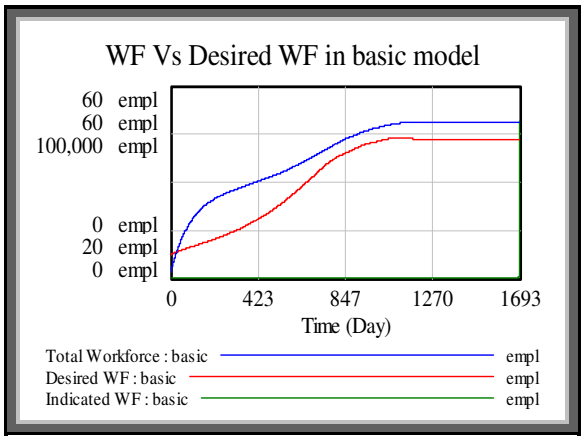


Figure 8: WF and Desired WF in basic model

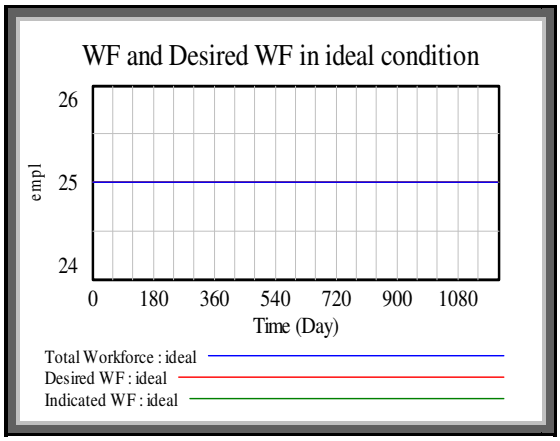


Figure 9: WF and Desired WF in ideal condition

Furthermore, when I simulate the basic model, from the Figure 8 we can see that the initial desired workforce is 25, while the real workforce in the model is just 2. There is a labor deficit from the first beginning. That is why the organization needs to hire more people according to the deficit.

So we can imagine that, in the ideal condition, if I set 25 as the initial value of *Total Workforce,* that is, equal to the initial desired workforce, then the organization does not need to hire more people. The labor deficit is zero all the time. The *Total Workforce* and *Desired WF* should be fixed in 25 through the whole project. Figure 9 shows the graphs in this certain condition. The graphs express the same story as my analysis.

After the making the validation of the model, I believe my model is correct.

4.2. Step 2: Adding working intensity and working overtime

After finishing the basic model, I move to the next step---adding overtime part in the model. It is another basic method to close the target-performance gap in performance dimensions addition to slipping the deadline.

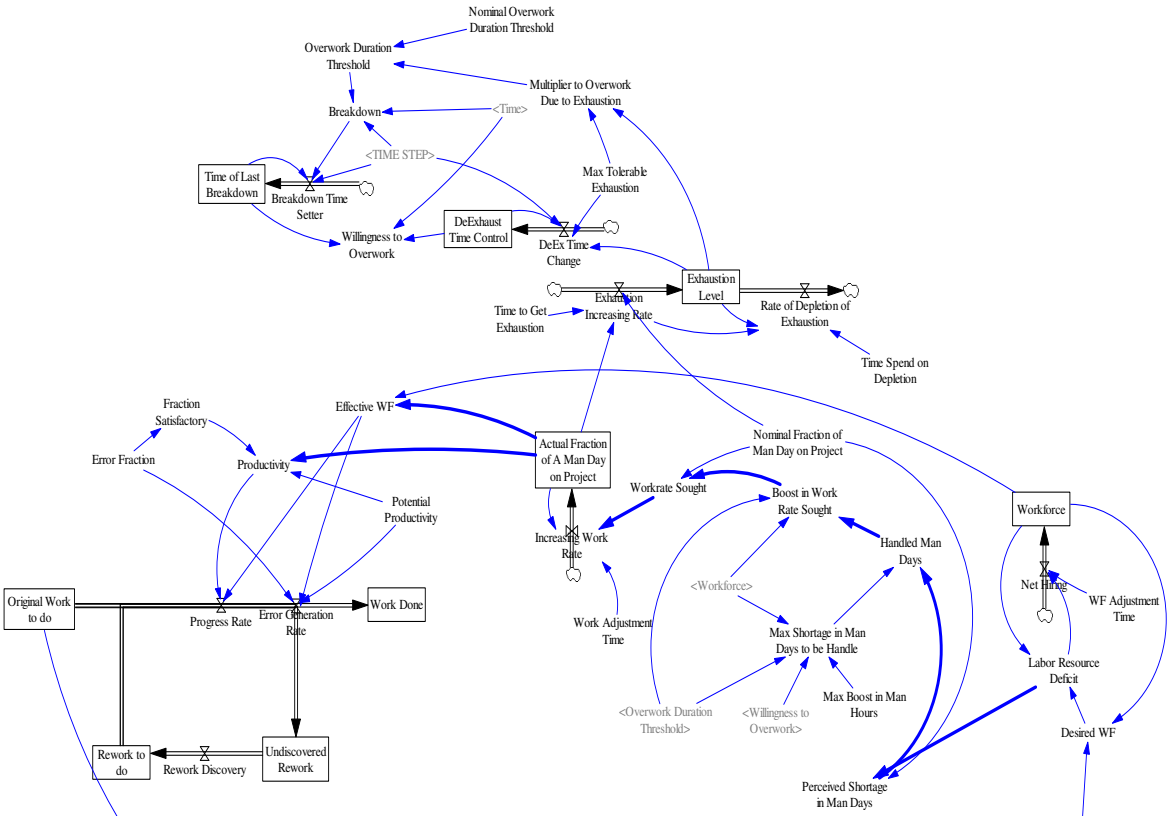


Figure 10: Model in overtime part

4.2.1. General description

In a project, labor resource deficit causes a shortage in man days, and then leads project progress behind schedule. In order to catch up with the schedule, employees boost the work rate by reducing slack and working overtime. So *Actual Fraction of Man Day on Project (AFMDP)* is more than the nominal value 0.6 in such situations. But when people work in above working hours, they get exhausted. Once the exhaustion levels due to overwork reaches the maximum tolerable exhaustion, which means that the overwork duration threshold is driven to zero, the employees don't want to work as overtime any more until their "Exhaustion level" is fully depleted. Figure 10 illustrates this part of the model.

In this step, there are two points we should pay attention. One is that *AFMDP* is represented as a stock in the model of this step, but in the basic model, it is a constant value 0.6. The other one is that *AFMDP* not only just influences the *Productivity* but also influences *Effective WF* in this model. Detailed analysis about *AFMDP* is given below.

4.2.2. The definitions of Effective WF and Productivity

Before I go deep into this model, we have to clarify what *Effective WF* and *Productivity* exactly mean in this model. I define *Productivity* as "How many tasks are done by individual employee per day of 8 hours". *Effective WF,* as the verbal sense, means effective workforce. The value of total workforce does not equal to the value of effective workforce. In this model, when employees are overworking, that is *AFMDP* is more than 1, *Effective WF* is bigger than total workforce.

4.2.3. Work intensity and overtime

As I mentioned before, in most projects, increasing work intensity and overtime work are caused by the schedule pressures. In real life, people like to spend some time on personal activities, such as chatting and drinking coffee. Consequently, I introduce the parameter *"Nominal Fraction of Man Day on Project" (NFMDP)* in the model, which is defined as the fraction of daily hours allocated to project-related work by the average full-time employee. I set the value 0.6 to the *NFMDP*. This means that in the absence of schedule pressures, a full time employee would allocate 0.6*8=4.8 hours to the project (assuming 8-hour day). When the project seems to be behind the schedule, the pressures increase. Employees tend to work harder and allocate more man-hours to the project by reducing the slack time to compensate for the perceived shortage and bring the project back on schedule. Since the average project-related working time is 4.8 hours in non-pressure condition, at most 8-4.8=3.2 hours per man-day can be gained by compressing slack time under schedule pressure. While sometimes the perceived shortage is too much, in addition to partially compressing their slack time, employees may also work overtime hours. For example, by working 12 hours a day at 80% efficiency, an employee would be allocating 9.6 hours to the project, thereby doubling the nominal 4.8 hours.

In the model, in order to implement *Working Slack-off* and *Overtime*, I use a stock-and-flow structure "*Actual Fraction of a Man Day on Project" (AFMDP).* The initial value is the value of *NFMDP*=0.6. As the inflow comes in, the value increases. This course includes both situations of increasing work intensity and working overtime. During the time of increasing *AFMDP* from 0.6 to 1, it is the increasing work intensity part in which *Productivity* would be affected. *Productivity* will increase following *AFMDP* until the value of *AFMDP* is up to 1, at which point *Productivity* reaches its maximum value in a constant fraction satisfactory level. Once increasing *AFMDP* beyond 1, people are overworking. In this working overtime period, the increasing value of *AFMDP* affects only *Effective WF* and *Productivity* will stay on the maximum value. From the analysis above we know that both work intensity and overtime parts have connection with *AFMDP*. In order to distinguish them, I use IF THEN ELSE function in the model's equations.

*Productivity = "IF THEN ELSE (AFMDP<=1, AFMDP *Fraction Satisfactory*Gross Productivity, Fraction Satisfactory*Gross Productivity)"*

*Effective WF = "IF THEN ELSE (AFMDP>1, Actual Fraction of Man Day on Project*Workforce, Workforce)"*

4.2.4. Fatigue

As the overtime work goes on, people are getting more and more exhausted. According to interviewees [13], there is a threshold beyond which employees would not be willing to work above normal. The nominal value of the threshold is 50. Once people start working harder to handle the shortage in man-days, their *Overwork Duration Threshold* decreases below the nominal value. Thus the *Overwork Duration Threshold* is formulated as a nominal value (e.g. 50) adjusted downwards by a multiplier. In my model, I formulate the *Multiplier to the Overwork Duration Threshold* due to *Exhaustion*.

Exhaustion is a level whose value reflects the level of exhaustion of the work force due to overwork. The rate at which this level increases needs to be a function of some measure of overwork, in ref. [12], Abdel-Hamid and Madnick provided a figure shown in Figure 11

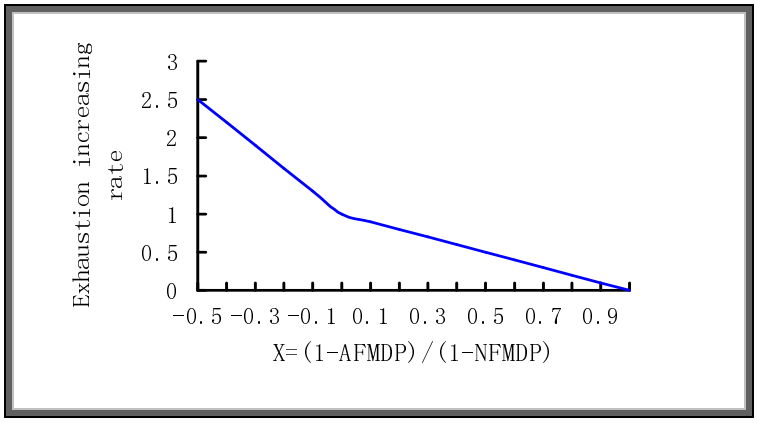


Figure 11: Exhaustion Increasing rate

The first thing we should note from figure 8 is that when *AFMDP* no bigger than *NFMDP,* that is X>1, the value of exhaustion increasing rate is 0. It means when people working on their normal pace, no exhaustion level produces. Second, when *1-AFMDP* bigger than 1, people just work in reducing their slack time. However, when *1-AFMDP* approaches 0 and move even to negative value, besides compressing slack time, people also work in overtime. That is why the curve increases faster for negative values of X.

Notice that when *AFMDP* is 1, the Exhaustion Increasing Rate is also 1. At such rate, each man-day contributes 1 to *Exhaustion Level*. After 50 days, the total exhaustion level would reach to 50, which is enough to drive *Overwork Duration Threshold* to zero. This level of

exhaustion level is called the *Maximum Toleration Exhaustion,* and this maximum value can be reached before 50 days if people are working even harder, in other words, *AFMDP* is bigger than 1. Conversely, if *AFMDP* is less than 1, the Maximum Toleration Exhaustion would be reached after 50 days. No matter when the maximum value is reached, *Overwork Duration Threshold* becomes to zero. This is accomplished by the formulation of the *Multiplier to the Overwork Duration Threshold* due to *Exhaustion* shown in figure 12.

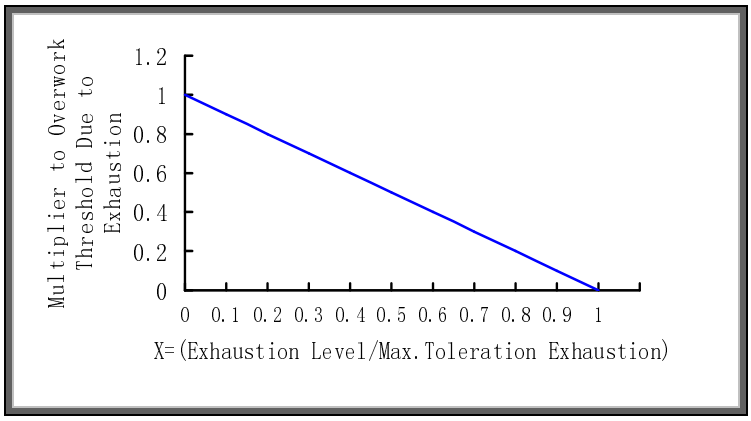


Figure 12: Multiplier to overwork due to exhaustion

Once the exhaustion levels due to overwork reaches the maximum tolerable exhaustion, which means that the overwork duration threshold is driven to zero, the employees don't want to work overtime any more until their "Exhaustion level" is fully depleted. They returns to a normal work rate where *AFMDP=NFMDP*. It is implemented in the model by the SWITCH variable *Willingness to Overwork* that can attain one of two values, zero or one. The value of *Willingness to Overwork* switches to zero whenever the *Overwork Duration Threshold* is driven down to zero and it remains at zero until the work force is fully "de-exhausted". Since *AFMDP* decreases from the top back to less than 1, it would affect the productivity again. The productivity would decrease during the depletion time.

4.2.5. Boost in work rate

In the model, labor resource deficit causes shortage in man days, and then leads project progress behind schedule. In such situations, the employees seek to boost their work rate to handle either *Perceived Shortage in Man Days* or the *Maximum Shortage in Man Days to be handled*, whichever is smaller. The smaller of the two values I call *Handled Man Days.* The percentage of *Boost in Work Rate Sought* equals the value of *Handled Man Days* divided by *Workforce and Overwork Duration Threshold*. And the difference between boosted work rate sought and AFMDP determine the increasing work rate. I set work rate adjustment time is 14 days.

4.2.6. Model validation

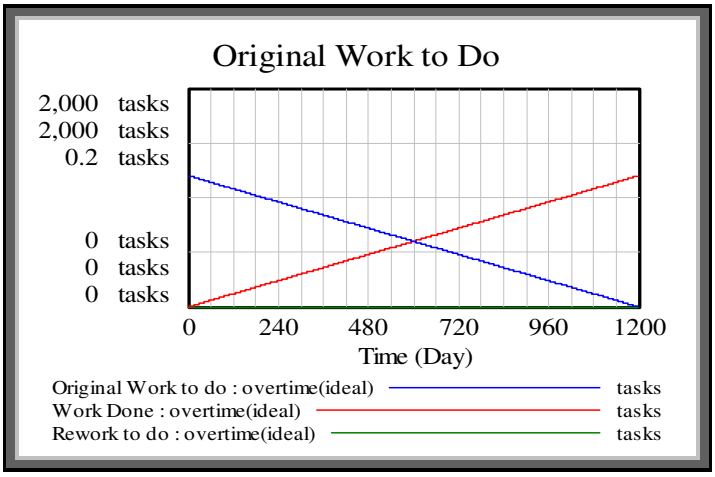


Figure 13: Project Progress in ideal condition in the model with overtime effect

As the same as the basic model, I put 1 in both *AFMDP* and *Satisfactory Fraction* in the overtime model to test it. The result graph shows in Figure 13. We can see that in the ideal condition the original works are finished on the day 1200 and the rework to do is 0 all the time. These behaviors match the analysis that I gave in the basic model.

Moreover, I also set the initial workforce value to 25 in the model, which is the initial value of desire workforce. In addition, because it is an ideal condition, which means *AFMDP* is 1 and *Error Fraction* is 0, productivity equals potential productivity, there is no hiring required and no labor resource deficit during the project. Therefore, the *Perceived Shortage in Man Day* should be nearly zero; *Handled Man Day* should be nearly zero, no *Boost in Work Rate*, no *Increasing Work Rate*. The *AFMPD* would stay on its initial value 1 all the time. Figure 14 shows my analysis.

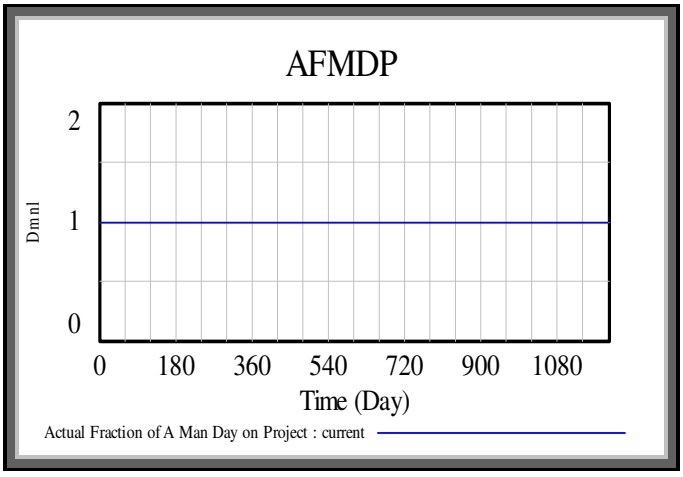


Figure 14 AFMDP in ideal condition

As I explained in previous section 4.2.4, when AFMDP is 1, *Exhaustion Increasing Rate* is

also approximately 1. At such work rate, each man-day contributes 1 to the *Exhaustion level*. After 50 such days, the *Exhaustion level* reaches a level of 50, which should be enough to drive the *Overwork Duration Threshold* to zero. That level of Exhaustion is termed the *Maximum Tolerable Exhaustion* level 50. In the ideal condition, because *AFMDP* is always 1, we should get the graph according to the assumption above.

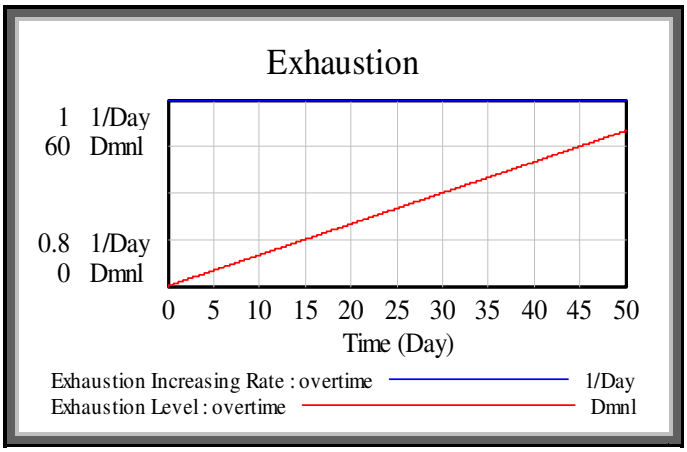


Figure 15: Exhaustion level and Exhaustion Increasing Rate in ideal condition

Figure 15 gives the result graphs. In order to see the values of graphs clearly, I put the final time is 50 days here. From the figure, we can see that both *Exhaustion level* and *Exhaustion Increasing Rate* match the assumption.

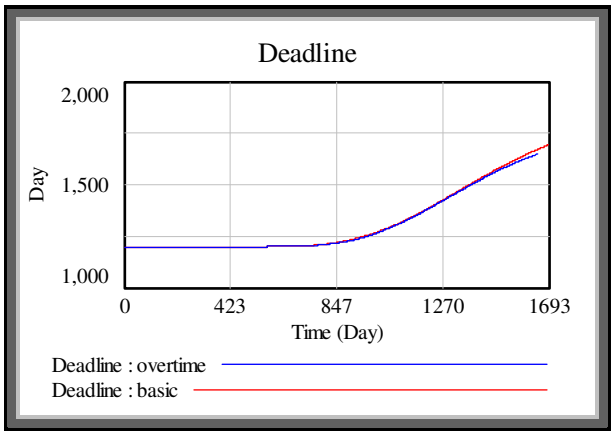


Figure 16: Comparison of deadline in two steps

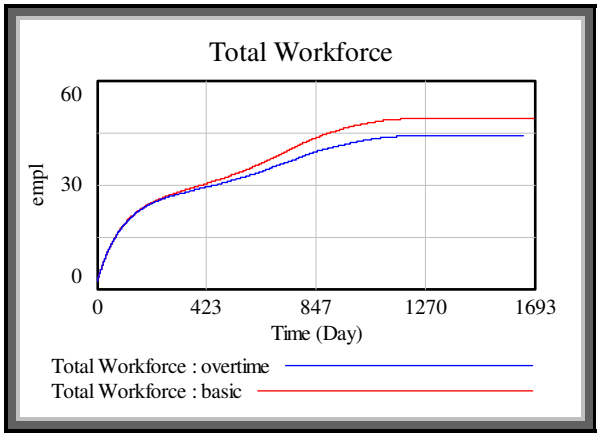


Figure 17: Comparison of total workforce in two steps

Because *AFMDP* is boosted several times in overtime model, it leads real productivity and effective workforce in this step bigger than the ones in basic model. Besides, in this step I haven't considered the ripple effects to the quality, so the value of error fraction is 0.3 constantly. Consequently, the projects would be finished earlier in the condition with working overtime than without, which means the simulation result of deadline in step 2 should be less than step 1. Figure 16 shows the comparison of the simulation results. We can see that, the

project does finish earlier in working overtime condition than basic one. The same as total workforce, since productivity increases in overtime step, less people are desired, the total workforce is less than the one in basic model. See Figure 17. Every simulation figure above fits the analysis I give. I believe that my model is so good so far.

4.2.7. Behavior comparison with the basic model

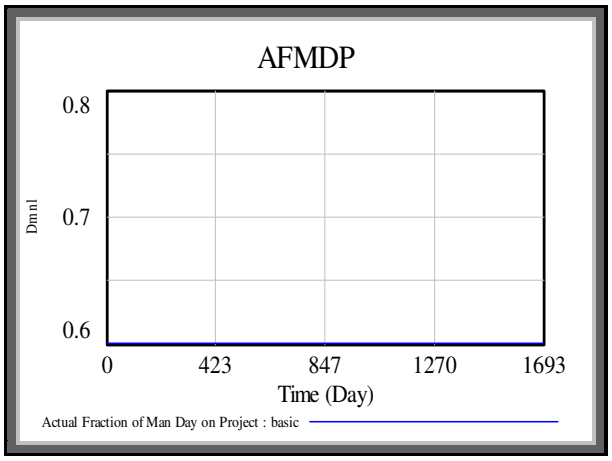


Figure 18: AFMDP in basic model

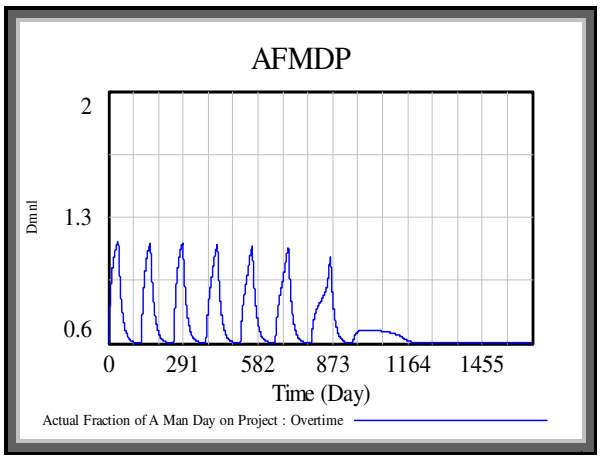


Figure 19: AFMDP in the model with overtime

Figure 18 and Figure 19 show the *AFMDP* in both basic and overtime model. We can see that the curve is boosted several times in overtime model, while in the basic model, the curve stays at 0.6.

In the basic model, the *AFMDP* is set to a constant value 0.6. In the overtime model, the *AFMDP* is a level and it can be boosted when the employees feel schedule pressure. 0.6 is just its initial value; it increases during the slack off and even up to 1.2 in the overtime. But when employees are exhausted to work above working hours, which means that the overwork duration threshold is driven to zero, *AFMDP* returns to normal work rate 0.6. And it will be boosted again when the exhaustion level is eventually depleted. After 900 days, as net-hiring and progress go, the labor deficit becomes small, so employees don't need to reduce slack nor work overtime. *AFMDP* is back to normal 0.6.

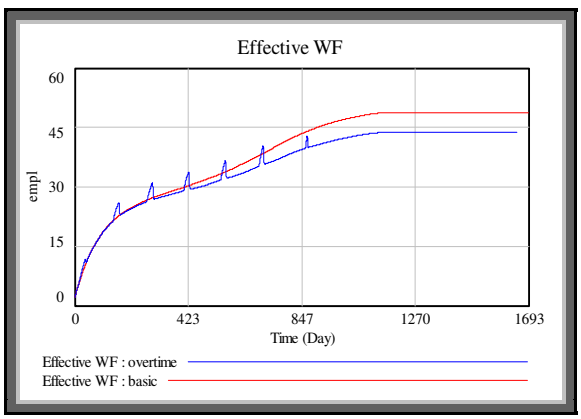


Figure 20: Comparison of Effective WF in two steps

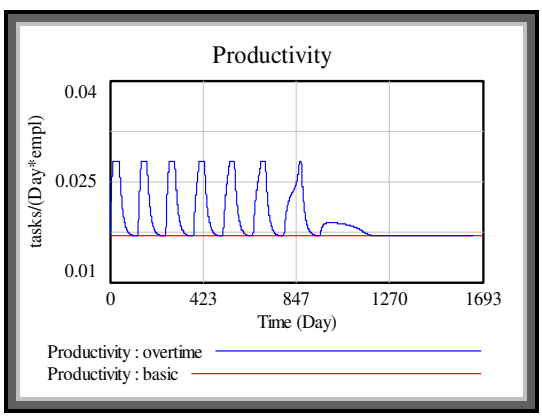


Figure 21: Comparison of Productivity in two steps

In the overtime model, increasing *AFMDP* from 0.6 to 1 affects *Productivity*, and when *AFMDP* is beyond 1, *AFMDP* affects *Effective WF.* Figure 20 and 21 shows comparison figures of these two variables in two steps. The curve of *Productivity* follows the curve of *AFMDP* when its value is under 1. When *AFMDP* is more than 1, the value of Productivity remains the maximum value, while the Effective WF is boosted. From Figure 20, we also see that the curve of *Effective WF* in overtime step is lower than the one in the basic step. That is because in the model,

Effective WF = "*IF THEN ELSE (AFMDP*>1, *Actual Fraction of Man Day on Project*Workforce, Workforce)*"

Less total workforce leads to lower effective workforce in overtime step.

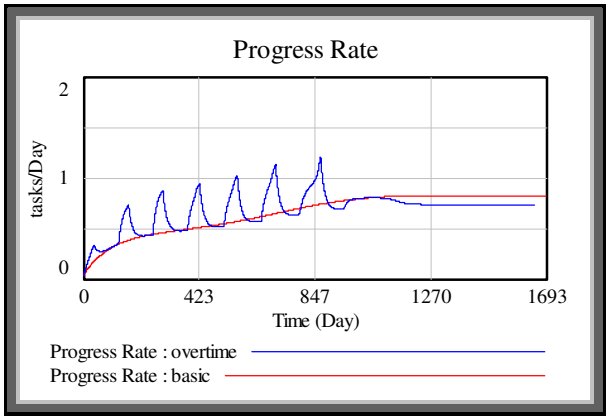


Figure 22: Comparison of Progress Rate in two steps

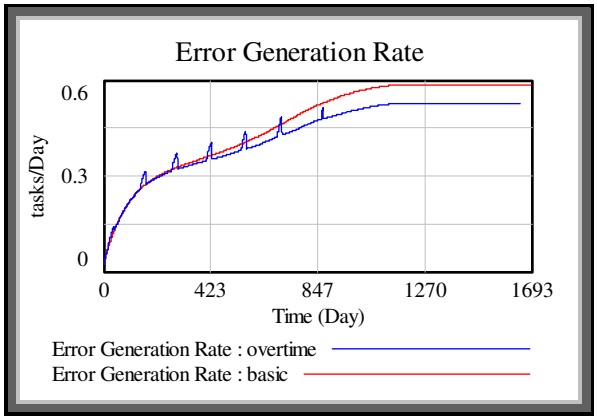


Figure 23: Comparison of Error Generation Rate in two steps

Figure 22 and Figure 23 show the *Progress Rate* and *Error Generation Rate* in both basic and overtime model. We can see that the *Progress Rate* curve is boosted sometimes in overtime model comparing with the smooth curve in basic model.

In the model, I define that *Progress Rate* equals *Productivity*Effective WF,* and *Productivity* and *Effective WF* are affected by *AFMDP* in different work rate. According to the statement I gave previously, the *AFMDP* is boosted when necessary sometimes, *Productivity* and *Effective WF* changes as consequence, which leads the result in the graphs of overtime model —the *Progress Rate* is boosted too when the *AFMDP* is boosted.

About the *Error Generation Rate,* when *AFMDP* is boosted, people are under pressure, working fast and getting exhausted, so it is easier to make mistakes. That is why the *Error Generation Rate* is also boosted a bit when the *Progress Rate* is around the peak in the graph.

4.3. Step 3: Ripple effects

This step is to add factors of ripple effects which are primary impacts of project control on rework and productivity in the model. These effects typically reduce productivity or quality. Figure 24 illustrates the part of the model with ripple effects.

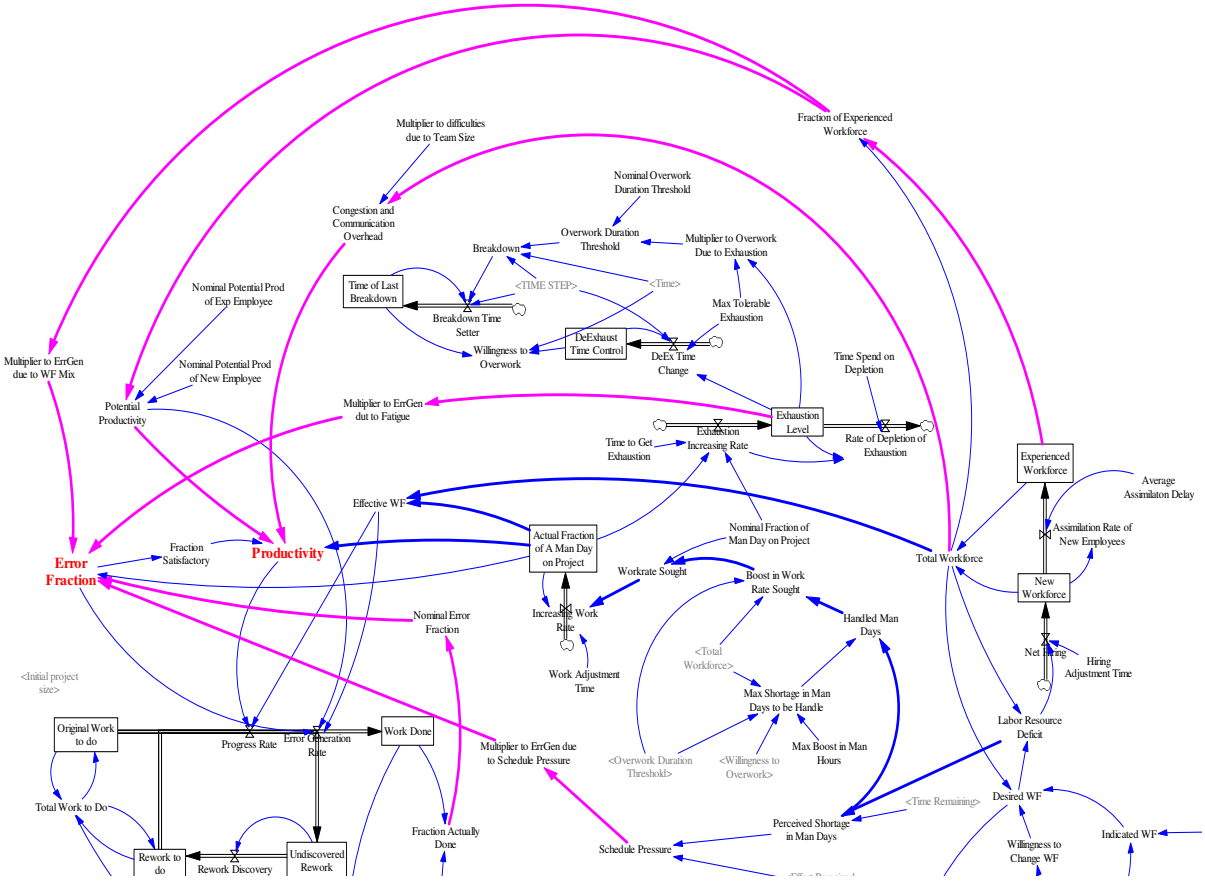


Figure 24: model of ripple effects

4.3.1. Factors influencing productivity

From the analysis above, we know that the value of the *Actual Fraction of a Man-Day on Project* becomes an important determination of actual productivity. It represents the losses in productivity due to motivational factors. But it is not the only determinant; there are also additional losses in productivity due to other factors.

4.3.1.1. Experience

In a project, because error fraction and actual effort are not perfect, the desire workforce is more than total workforce. The labor deficit is generated, so the company would hire more people during the project. In previous steps I assumed that all employees have the same levels in experience. But in real life, those new hired people, no matter they are from outside the

company or transferred from other teams inside the company, are inexperienced workforce in this project. I call them *New Workforce*, and they have to take an average assimilation delay to become *Experienced Workforce.* Therefore, now we get the *Total Workforce* in the model, which equals *New Workforce+ Experienced Workforce.* I assume that the initial 2 employees are all experienced ones, and the average assimilation delay is 80 days in the model.

Normally, because of orientation, the newly hired employees are, on average, less productive than the experienced ones. Hence, total potential productivity changes as consequences. The equation is

*Potential Productivity=Fraction of Experienced Workforce*Nominal Potential Prod of Exp Employees+ (1-Fraction of Experienced Workforce)*Nominal Potential Prod of New Employees.*

In the model, I assumed that the nominal potential productivity of new employees is half of the value of experienced ones.

4.3.1.2. Communication Losses

Human communication is an essential component during a project. It is necessary that each individual spend part of his time communicating with each of the other team members, but it does constitute an overhead too. The increase of the project team leads to the increase of individual communicating time, increase of the amount of documentation and the increase of other additional work. This results the average team member's productivity to drop below his nominal productivity. For example, if the *AFMDP* is 0.6, then a full-time employee allocates on the average 0.6*8=4.8 hours to the project. If the project communication overhead consumes 25% of the allocation, then the average productive fraction of the Man-Day would be 0.6*(1-0.25)=0.45, and the real average working time of a full-employee would be 0.45*0.8=0.36 hours as consequence.

It is widely held that communication overhead increases in proportion to n^2, where n is the size of the team [14]. In the model, I introduce a parameter *Multiplier to difficulties due to Team Size,* and its value is *f*, so I get the equation $f*Workforce^2$=*communication and congestion overhead.* Because the size of projects is different, ways to decide communication overhead are diverse too. In this case, I assumed that when the number of a team is 30, communication overhead is approximately 10%. Therefore, according to the equation above, I get the value of *Multiplier to difficulties due to Team Size f* is $0.1/30^2=10^{-4}$. So *communication and congestion overhead*= $10^{-4}*Workforce^2$. I assume this value is constant in the model.

Consider the case where n=40; communication overhead is approximately 16%. If *AFMDP* is 0.6, and 16% of it will be lost due to communication overhead. In other words, multiplier to

productivity due to communication losses is 0.6*(1-0.16) =0.504, which means that real *Productivity* is just about 50% of the *Potential Productivity*.

After adding ripple effects in the model, the equation of *Productivity* is:

*Productivity=IF THEN ELSE (Actual Fraction of A Man Day on Project<=1, Actual Fraction of A Man Day on Project * (1-Congestion and Communication Overhead) * Fraction Satisfactory*Potential Productivity, Fraction Satisfactory*Potential Productivity*(1-Congestion and Communication Overhead))*

4.3.2. Error Fraction

As I mentioned above, in real life, original works are not all successfully done in the first pass. In the development of project involves a series of production activities where the opportunities for interjection of human fallibilities are enormous [15]. Because of human inability to perform with perfection, quality problems come out and they play a very important role in the whole system. So now I start to discuss more details about quality issue.

4.3.2.1. Nominal Error Fraction

The first set of factors which affect the *Error Fraction* includes organizational factors (e.g., the use of structured techniques [16], and the quality of the staff [17]) and project factors (e.g., complexity, size of system, language). Even though such factors are different from project to project, they remain invariant during the life of a single project. The collective effect of all such factors represented in the model as a single nominal variable, the *Nominal Error Fraction.* Since this nominal variable represents different types of errors in the model, it is not a constant value but changes over the project development. In order to formulate the *Nominal Error Fraction* according to my case, I assumed a function shown in Figure 25.

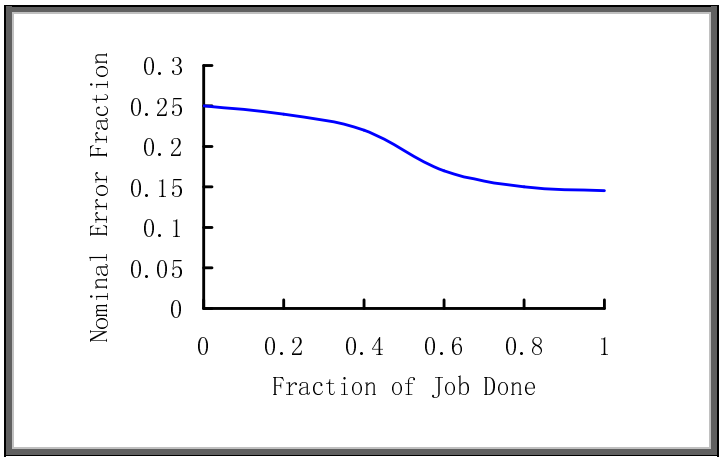


Figure 25: Nominal Error Fraction

From the figure 25 we can see that the *Nominal Error Fraction* is at the biggest value 0.25 in

the beginning of the project. The value decreases along the increase of the fraction of job actually done. The smallest value 0.145 is got when the project is fully finished.

4.3.2.2. Workforce Mix and Schedule Pressure

Nominal errors just represent the cumulative error generation from the organizational and project factors. Such factors remain invariant during the project. There is another set of factors that play a dynamic role during the project development: The workforce-mix and schedule pressure.

4.3.2.2.1. Impact of Workforce Mix

As I discussed in the chapter 4.3.1.1, the work force comprises two types of employees, newly hired and experienced ones. Newly hired employees have to pass a training period to become experienced ones. This kind of orientation process improves new employees in both the social and technical aspect of the project.

Newly hired employees are not only less productive on average, but also make more errors than the experienced counterparts. In the model I assumed that new employees make twice as many errors as the experienced ones. To model this, I formulate the *Multiplier to Error Generation Due to Workforce Mix* as a function of *Fraction of Experienced Workforce*. See Figure 26.

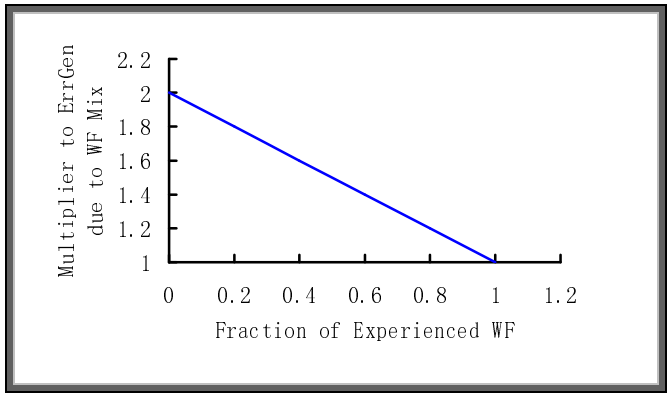


Figure 26: Multiplier to Error Generation Due to Workforce Mix

When the workforce consists only of experienced staff, the value of the multiplier is set at 1, since it would still have nominal error generation. As the fraction of new employees increases, the multiplier increases linearly until it reaches a maximum value 2, which means that the workforce are all new hires.

4.3.2.2.2. Impact of Schedule Pressure

Schedule pressure is a very important parameter in the model. Recall that employees boost

their working rate by reducing slack time and even working overtime in order to catch up the schedule. In the model, schedule pressure is defined as:

Schedule Pressure= Perceived Shortage in Man Days/Effort Perceived Remaining

Schedule pressure can also increase the number of errors [18]. People under time pressure do not work better, they just work faster. Schedule pressure also increases the "anxiety levels". When the deadline approaches, employees become more anxious, and tend to make more mistakes.

The effect of schedule pressure on error generation in the model is shown in Figure 27. Under the nominal condition there are no schedule pressures, and the multiplier assumes a value of 1. As the schedule increases, the multiplier increases indication higher error.

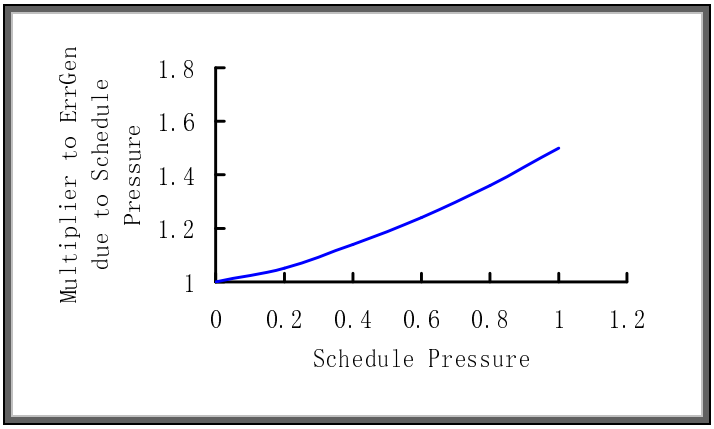


Figure 27: Multiplier to error generation due to schedule pressure

There are two points we should pay attention here. One is that schedule pressure determines work intensity, so, in the model, I use *Schedule Pressure* to formulate the effects of "working slack-off" on error generation. The other point is that schedule pressure also often results in overlapping which is another significantly factor to increase the chance of errors. It is the secondary effect of schedule pressure. That is why error generation can increase as much as 50% under influence of schedule pressure according to the figure. I will discuss details about overlapping in the following chapters.

4.3.2.3. Impact of Fatigue

From the discussion of *Overtime* part in last chapter, we know that sometimes in order to catch up the schedule, employees would increase work intensity or even work overtime. That causes exhaustion, or in another word, fatigue. It is the primary effect of *Overtime*. However, through fatigue, there are also the secondary effects of *Overtime* in the system. Imagine if you are overworking and very tired, do you have much energy and good attitude to make things

perfect, or do you just want to finish them as soon as possible? I think the latter one is the more common answer. The same as in this case when employees are getting exhausted, they would pay less attention in the quality, and the error generation increases as result.

Figure 28 shows the function that I assume how exhaustion level can affect quality. When *Exhaustion level* is 0, which means employees work in nominal condition and no effect is on error generation due to fatigue, the multiplier is 1. As the exhaustion levels increases, the multiplier increases. The maximum multiplier is 1.23.

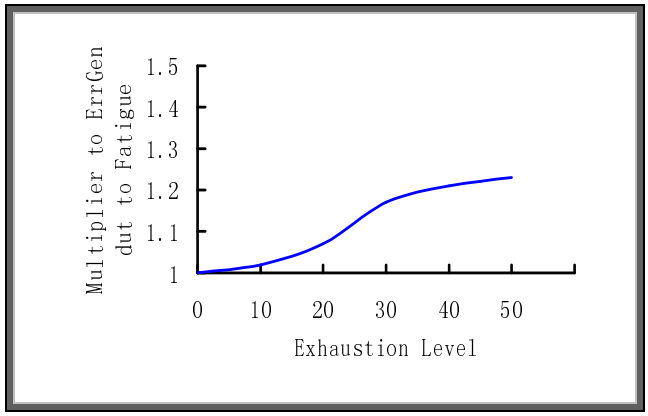


Figure 28: Multiplier to Error Generation Due to Fatigue

Notice that fatigue only affects error generation during working overtime, so in the model, I formulate that *Multiplier to Error Generation due to Fatigue* can only be used when the value of *AFMDP* is bigger than 1. I use IF THEN ELSE function to implement it.
After adding ripple effects in the model, the equation of *Error Fraction* is:

*Error Fraction= IF THEN ELSE (Actual Fraction of A Man Day on Project>1, Nominal Error Fraction*Multiplier to ErrGen due to WF Mix*Multiplier to ErrGen due to Schedule Pressure*Multiplier to ErrGen due to Fatigue, Nominal Error Fraction*Multiplier to ErrGen due to WF Mix*Multiplier to ErrGen due to Schedule Pressure)*

In addition, since *Error Fraction* would increase during working overtime because of secondary effect of *Overtime*, *Fraction Satisfactory* would decrease and *Productivity* would decrease too as consequence. Therefore, actually, fatigue not only affects the error generation but also the real productivity during overtime.

4.3.3. Model Validation

In order to test if the model so far is correct, I use two steps to do validations.

First, I don't consider the effect of congestion and communication difficulties on *Productivity*,

in other words, the *Congestion and Communication Overhead* is 0. Under this prerequisite, I assume that *Nominal Error Fraction* is 0 and the value of *Nominal Fraction of Man Day on Project* is 1. Besides, I set the initial value of *Experienced Workforce* to 25 which is also the initial value of *Desired Workforce*. The simulation results show in following figures.

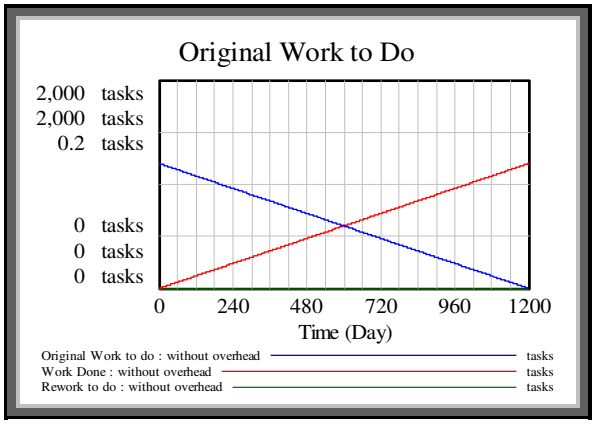


Figure 29: Progress in without overhead condition

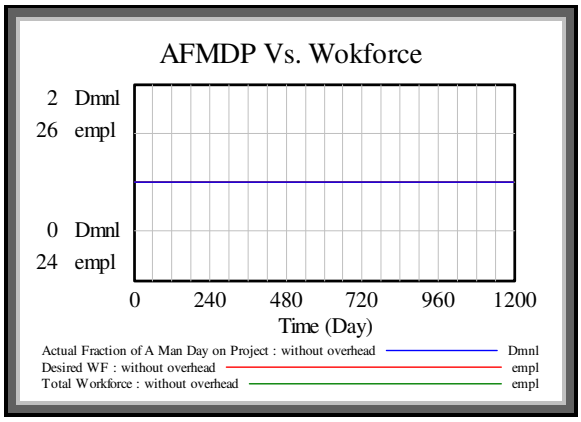


Figure 30: AFMDP vs. Workforce in without overhead condition

From Figure 29 and 30 we can see that the whole project would finish in exactly 1200 days, there is no labor deficit and *AFMDP* remains at value 1 all the times. I have analyzed those behaviors in the validation part of last two steps. But still, I want to say more here. In this model, in order to get the get the ideal without-delay result, in addition to set the error fraction to 0 and *NFMDP* to 1, I also have to make the initial total workforce value equal to initial desire workforce value 25. While in the previous steps of models, the second condition is not necessary. The reason why the project can be finished without any delay is that *Productivity* equals *Potential Productivity* and the value has to be fixed. This condition can be reached just by setting the error fraction and *NFMDP* to 0 and 1 in previous steps. But in this step, because I consider the different productivities between experienced employees and the newly hired employees, hiring new people would influence the *Productivity*. So in order to keep the value of *Potential Productivity* constant, I have to keep the value of labor deficit at 0 from the beginning. The same as *Congestion and Communication Overhead,* I don't consider it here because the team size determines the overhead and the overhead would cause productivity to be unequal to potential productivity.

From the figures and the discussion above, I can say that the model is correct without considering the effect of congestion and communication difficulties. Now, I will consider this factor in the model to see what is going to happen in the behavior mode.

I still set *Nominal Error Fraction* to 0 and *NFMDP* to 1. The initial value of *Experienced Workforce* is also still 25, so there is a initial *Congestion and Communication Overhead* = 25*25*0.0001=0.0625. This overhead results *Productivity* to be unequal to *Potential Productivity*. Even though the difference is very small because only *Congestion and*

Communication Overhead affects productivity, *Desired Workforce* would still become bigger than *Total Workforce* regularly. Then a small *Labor Deficit* comes out, and company need to hire new people as consequence, but not much. Figure 31 shows the simulation result of *Workforce* in this condition.

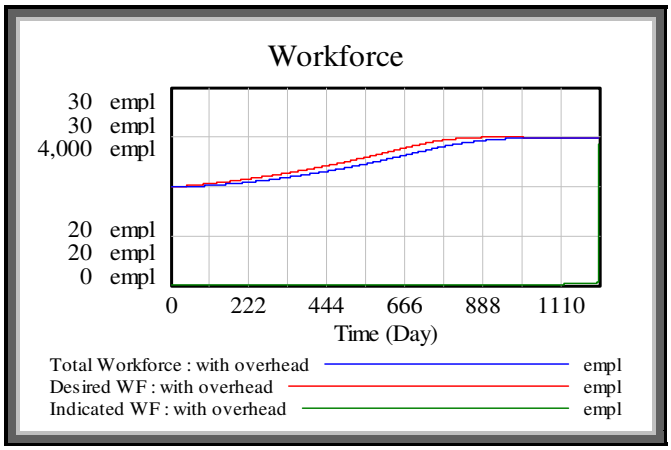


Figure 31: Workforce in with overhead condition

From the figure, we can see that both *Desire Workforce* and *Total Workforce* begin with the number of 25. The numbers increase slowly during the project, and the value of *Desired Workforce* is always slightly bigger than the value of *Total Workforce*. Finally, the both numbers end up around 27. The story shown in the figure matches the analysis I give above.

Now I continue my story in this condition. Recall the reason why *AFMDP* is always 1 in the "without overhead" condition. That is because there is no labor deficit through the project. No labor deficit means no *Perceived Shortage in May Days*. I formulate the value of *Handled Man Days* equals the minimum value of *Perceived Shortage in Man Days* and *Max Shortage in Man Days to be Handled*, so even though the value of *Max Shortage in Man Days to be Handled* is not 0 before the *Overwork Duration Threshold* is driven to 0, the *Handled Man Days* would still be 0 all the time. If no *Handled Man Days,* then no *Boost in Work Rate Sought* and *AFMDP* remain at 0. But in "with overhead" condition, since it has a small labor deficit in the project, there is a perceived shortage in man-days. Hence, the value of *Handled Man Days* is not 0 anymore at the beginning before overwork duration threshold is driven to 0. This would lead the boost in work rate sought, and then influence the *AFMDP*. But after the overwork duration threshold is driven to 0, the value of *Handled Man Days* turns to 0*,* no boost in *Work Rate Sought* and the *AFMDP* returns to the value of *NFMDP* (which is 1). Furthermore, because *NFMDP* is 1, *AFMDP* is always more than 1, the exhaustion level keeps rising, and the value of *Overwork Duration Threshold* remains once it is driven to 0. Therefore, the *AFMDP* remains at 1 during the rest of the project.

Figure 32 shows the simulation result of *AFMDP* in "with overhead" condition. We can see that *AFMDP* is boosted at the beginning of the project, then returns to 1 and remains at 1

during the project. This behavior matches my analysis too.

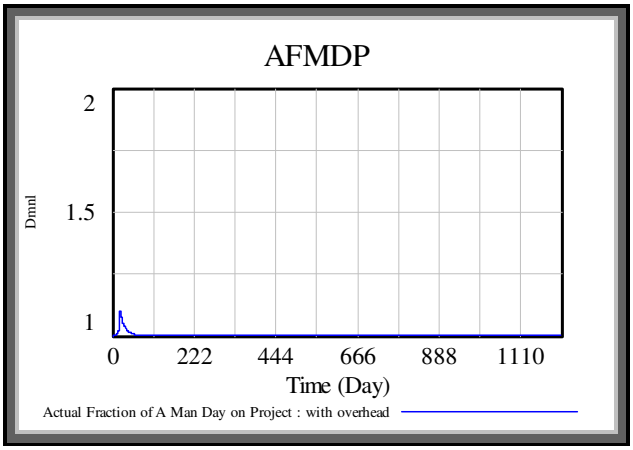


Figure 32: AFMDP in with overhead condition

Compared with "without overhead" condition, the completion date in "with overhead" condition should have a delay since *Productivity* less than *Potential Productivity*. Figure 33 illustrate the Deadline of both conditions. We can see that in "without overhead" condition, the project would finish on the day of 1200, while in "with overtime" condition it uses 1221 days to finish. It makes sense.

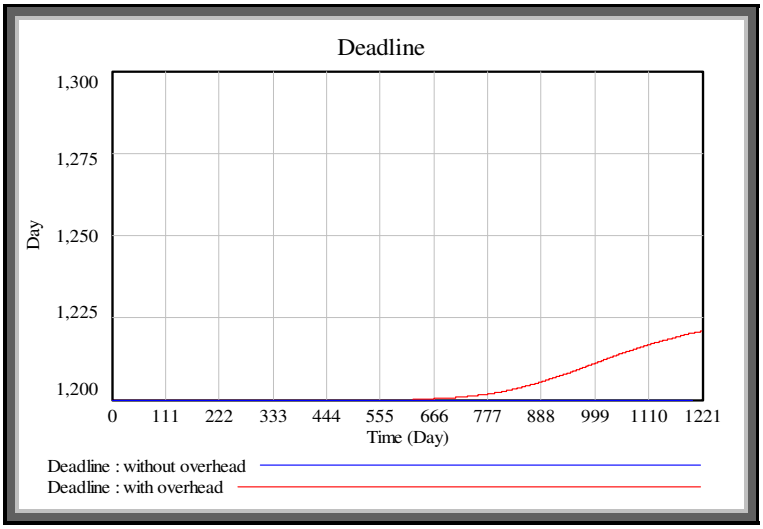


Figure 33: Deadline in with-overtime condition vs. deadline in without overtime condition

According the analysis above, I believe that my model is correct so far.

4.3.4. Behavior comparison with the "overtime" model

After adding the more detailed effects on productivity and error generation, the simulation behaviors are different from the last "overtime" step. Next, I will compare behaviors between last "overtime" step and the current step to see what have changed.

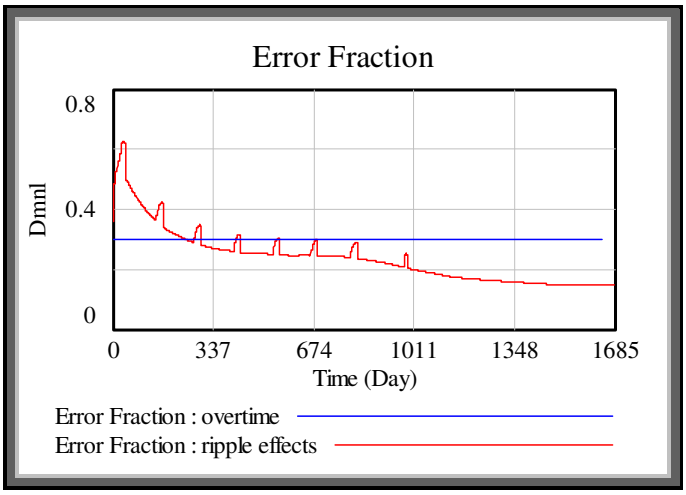


Figure 34: Comparison of Error Fraction in overtime model and ripple effects model

Figure 34 show *Error Fraction* in both overtime step and current ripple effects step. In overtime step, I just set the value of *Error Fraction* is 0.3. But in the current step, because I consider more factors which can influence quality, I get a new curve of *Error Fraction*. First, there is a nominal value of error fraction during the project. It can be seen as the basic *Error Fraction*. Then many multipliers which are effects from schedule pressure, workforce levels, and fatigue influence this curve. Notice that there are several boosts in the curve. That is because fatigue only influence error fraction when working overtime. In other words, *Multiplier to Error Generation due to Fatigue* only be used on determining the value of *Error Fraction* when *AFMDP* is bigger than 1. Therefore, the boosts in *Error Fraction* become when people are working overtime through the project.

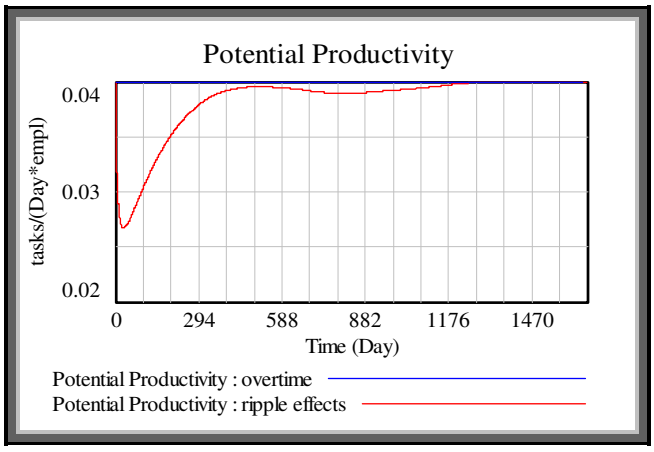


Figure 35: Comparison of Potential Productivity in overtime model and ripple effects model

The behaviors of *Potential Productivity* in overtime step and ripple effect step are shown in Figure 35. In overtime step, I set 0.4 to the value of potential productivity and it remains at this value because of no effect on it. While in the current step, I involve workforce level in the model. Different work levels lead different productivity. I assume that new employees are less

productive than experienced employees, so potential productivity is affected as new people are hired. From the simulation graph in ripple effect model, we can see the potential productivity is 0.04 at the beginning because all employees are experienced ones. The curve drops quickly at the beginning because there are only 2 people when the project starts, and it takes 80 days to become an experienced employees from a rookie. Therefore, the fraction of experienced employees drops quickly at the beginning of the project, and the *Potential Productivity* drops as consequence. After sometime, when more and more new employees finish the oriental phase to become experienced ones, the *Potential Productivity* increases back.

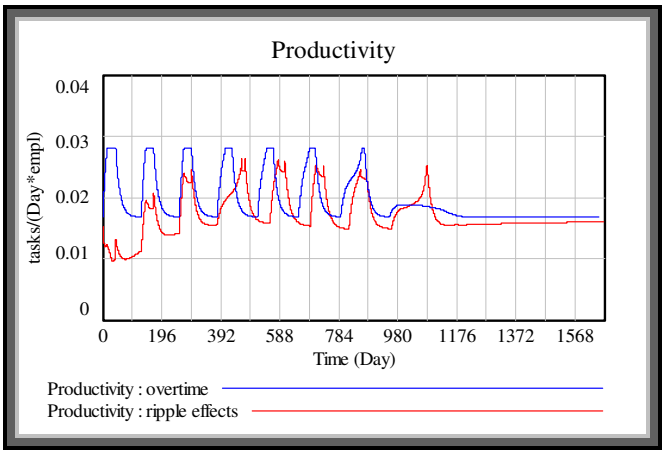


Figure 36: Comparison of Productivity in overtime model and ripple effects mode

Potential Productivity affects *Productivity.* Figure 36 show the *Productivity* in both steps. Compared with the stable curve in overtime step, we can see that in current step, productivity is small at the beginning and increases later because of the behaviors of potential productivity. Besides potential productivity, we have to consider another important effect on productivity. That is *Fraction Satisfactory*. In the model, I formulate *Fraction Satisfactory= 1-Error Fraction,* so the value of *Fraction Satisfactory* changes following *Error Fraction*. Figure 37 shows the behavior of *Fraction Satisfactory* in ripple effects step.

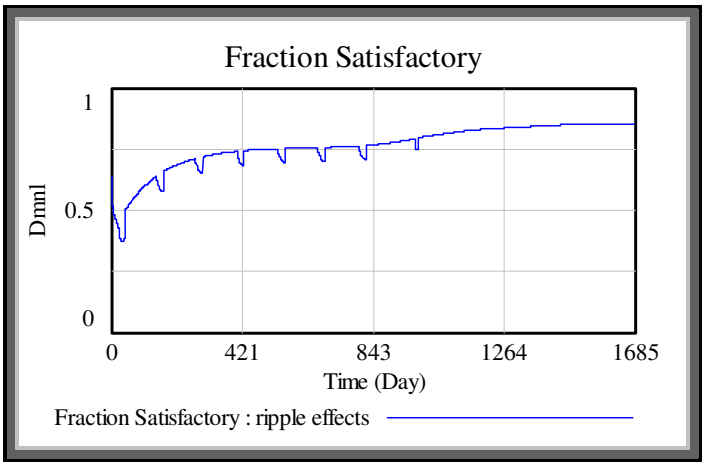


Figure 37: Fraction Satisfactory in ripple effects step

From the figure, we can see that although the trend of fraction satisfactory is increase during the project, it has many valleys in the curve too. The boosts in *Error Faction* when *AFMDP* is more than 1 cause those valleys. And these valleys are the reason why in the behavior curves of *Productivity* it always has drops around the peak value. This result shows that fatigue increases error generation and decreases productivity when people are overworking

4.4. Step 4: Knock-on effects

Knock-on effect is the effect of ripple effect and the secondary effect of policy resistances which are used to improve schedule performance of the project. Four knock-on effects feedback loops are represented in ref [5]. I will build these knock-on effects one by one in my model in this step.

4.4.1. Errors build errors

During the process of a project, the undiscovered errors do not just remain there until it is detected and corrected. They are active and would generate more and more errors in the system. For example, in the software project, an undiscovered design error can cause further additional errors in code.

In ref. [12], Abdel-Hamid and Madnick gave a proposed theory of the error reproduction progress in software project. They assumed that undiscovered errors will become either "Active Errors" that can reproduce errors in the further process, or "Passive Errors" that remain inactive in the system. I introduce Abdel-Hamid's proposal in my model. Figure 38 shows the model diagram of this "Error build errors" part.

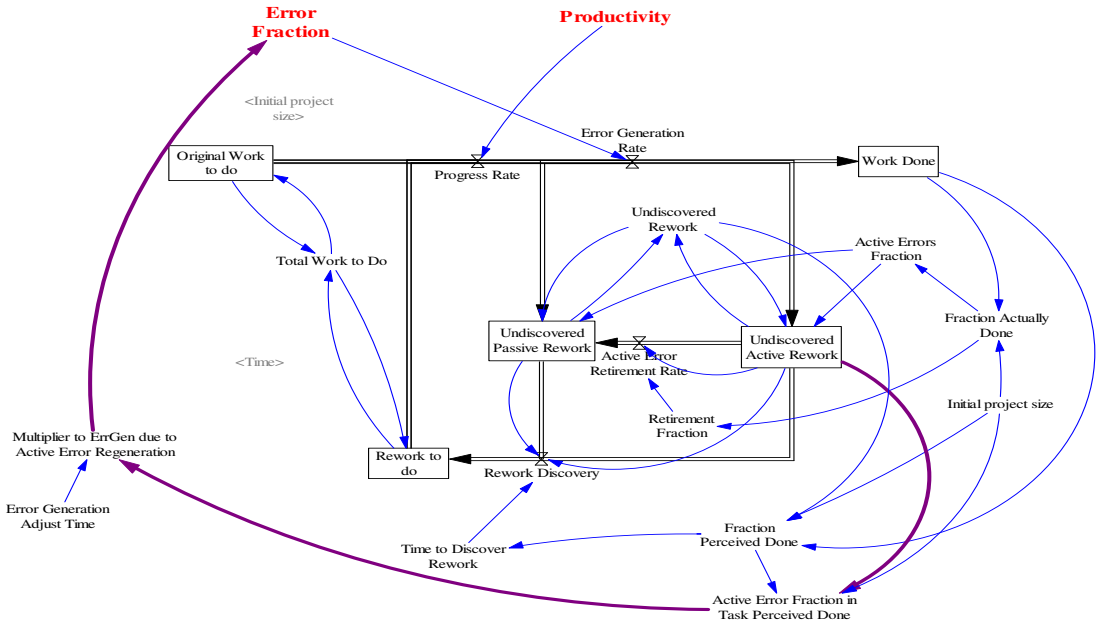


Figure 38: Errors build errors

In this step of the model, I divide the stock *Undiscovered Rework* into two stocks which are *Undiscovered Active Rework* and *Undiscovered Passive Rework*. Hence the unfinished tasks go to both stocks *Undiscovered Active Rework* and *Undiscovered Passive Rework* instead of one stock *Undiscovered Rework* through *Error Generation Rate*. Because projects are worked on top-down, in the beginning of the project, most errors are high level and active errors. As the development proceeds to lower levels, the fraction of active errors goes in the reverse direction since the errors are more localized. Therefore, the assumptions on how the mixture of active and passive reworks changes in error generation rate over the project life can be made. Figure 39 shows the assumption of the active error fraction in the error generate rate. Therefore, I can get the equations:

$(Error\ Generation\ Rate)_{Active}$=*Error Generation Rate*Active Errors Fraction*
$(Error\ Generation\ Rate)_{Passive}$=*Error Generation Rate*(1-Active Errors Fraction)*

In order to formulate the rework discovery outflow of *Undiscovered Active Rework* and the rework discovery outflow of *Undiscovered Passive Rework* separately, I take the same way as I used to distinguish the two different *Progress Rate* and *Error Generation Rate* in the basic model step—Proportion. It is the possible way that project managers could choose in a real project. I calculate the proportion of and *Passive Undiscovered Rework* in the whole *Undiscovered Rework* backlog. And I assume that this proportion is the same as the proportion in *Rework Discovery*, which are shared by *Active Undiscovered Rework* and *Passive Undiscovered Rework.* So I get the equations:

$(Rework\ Discovery)_{Active}$= *Rework Discovery*Undiscovered Active Rework/ (Undiscovered Rework+0.001)*
$(Rework\ Discovery)_{Passive}$=*Rework Discovery*Undiscovered Passive Rework/ (Undiscovered Rework+0.001)*

In the equations, since the *Undiscovered Rework* is 0 at the beginning, I add 0.001 to avoid the denominator to be 0.

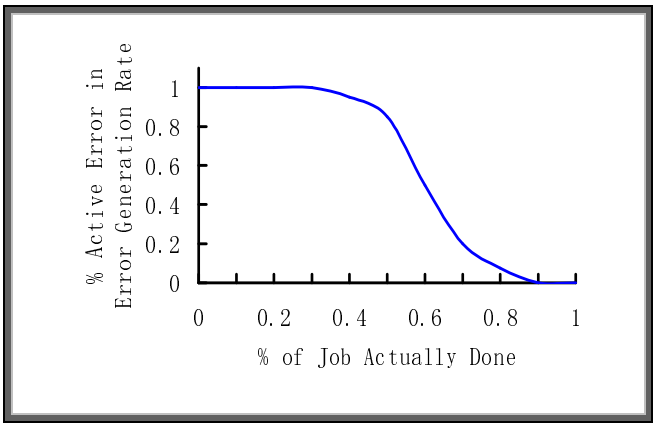


Figure 39: Active Error Fraction

As I stated above, the undiscovered active errors can produce new errors during the project. However, not all undiscovered active errors will continue to produce new errors until the end of the project. Some of them might stop after producing one or two generations of errors. In such conditions, those undiscovered active errors will actually become undiscovered passive errors. The rate is termed *Active Error Retirement Rate* in Figure 40. This rate is adjusted through *Retirement Fraction* which stands for the fraction of active errors that become passive every unit of time. At the beginning because active errors must generate at least one generation of new errors, no active errors will retire and become passive errors. *Retirement Fraction* remains at 0. As the progress going on, the project steps forward the stage of low level function modules. The errors in these low levels will not likely lead to further errors, so the error regeneration ability quickly decreases and the *Retirement Fraction* consequently increases quickly and reaches a value of 1 at the end of development. For example, in software projects, the first errors are active design errors and those design errors will produce code errors later. As the project progresses towards the coding level, less and less new errors will be produced, which means more and more active design errors retire to passive code errors.

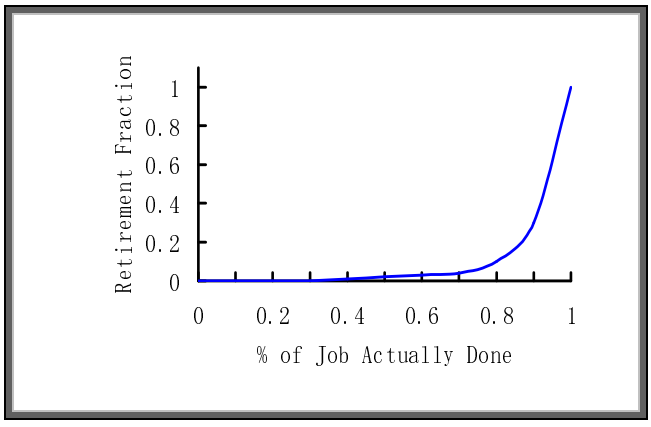


Figure 40: Retirement Function

After the statement above, I get the formulation of *Undiscovered Active Rework* and

Undiscovered Passive Rework:

Undiscovered Active Rework = (Error Generation Rate)$_{Active}$ -(Rework Discovery)$_{Active}$- Active Error Retirement Rate
*= Error Generation Rate*Active Errors Fraction-Active Error Retirement Rate - Rework Discovery*Undiscovered Active Rework/ (Undiscovered Rework+0.001). The initial value is 0.*

Undiscovered Passive Rework = (Error Generation Rate)$_{Passive}$ -(Rework Discovery)$_{Passive}$ + Active Error Retirement Rate
=Active Error Retirement Rate + Error Generation Rate (1 -Active Errors Fraction)-Rework Discovery * Undiscovered Passive Rework / (Undiscovered Rework+0.001) The initial value is 0*

Undiscovered Rework= Undiscovered Active Rework +Undiscovered Passive Rework

Active undiscovered errors regenerate new errors. In order to formulate the effect to *Error Fraction* through active error regeneration, I introduce a parameter called *Multiplier to Error Generation due to Active Error Regeneration* in the model. This multiplier is the function of *Active Error Fraction in Task Perceived Done*, which is formulated as existing undiscovered active rework divided by tasks perceived developed so far. Actually, the multiplier is the SMOOTH function of *Active Error Fraction in Task Perceived Done*. That is because the active errors would not affect other parts of works which are being done in parallel. They just can regenerate the errors through a finished task which is going to build on one another. Therefore, there is a delay before one active error builds other new errors. The average delay is set to 120 days here. The multiplier is a table function shown in Figure 41.

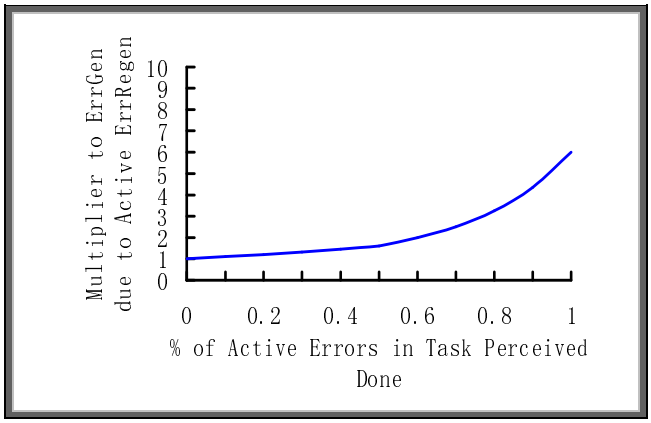


Figure 41: Multiplier to Error Generation due to Active Error Regeneration

The value of the multiplier is always 1 or above. That is because every undiscovered active

error will create at least one new error in the further progress. When it has no active errors in the system, the multiplier is 1. The value increases as the faction of undiscovered active errors in task perceived done. Studies have shown that systems were "characterized by the presence of 'error-prone modules' that show a high frequency of the system's total error content" [19]. For example, if there are 10 errors, they might be clustered in error group; but if there are 100 errors in the system, they are almost impossible still gathered in one error group. They would spread. Thus, higher fraction of undiscovered active errors in the task perceived done causes a wider distribution of errors in the system. In addition, because errors become less localized, they are hard to detect. Thus undetected active errors will have more time to generate new errors. That is why the value of *Multiplier to Error Generation due to Active Error Regeneration* is very big at high fractions of undiscovered errors in task perceived done.

4.4.2. Errors create more work

During the project progress, there are two types of new work effort required. The first is the one that I just discussed in the section 4.4.1. Undiscovered active rework results new further rework, so the extra effort is created to detect and correct those further new errors. This kind of extra work is still required in the existing project scope. Another form of required new work created is adding new tasks. Those new tasks create development activities beyond the previous project scope and it normally happens in the process of correcting discovered errors. For example, if an employee wants to fix a difficult discovered rework in the system, he might first go to check literatures or discuss with other people before really working on it. This fixing process with those extra actions takes more effort than doing the original work. In this section I focus on building "adding new task" part in my model. Figure 42 shows the conceptual diagram.

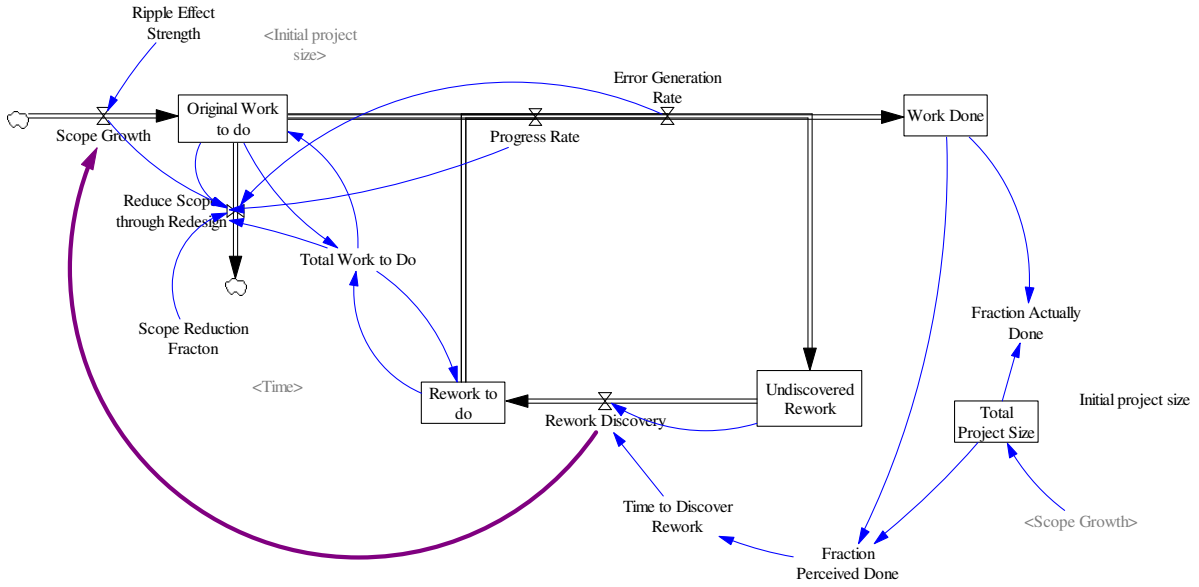


Figure 42: Errors create more work

As I mentioned before, adding new tasks often happens in the process of fixing discovered errors. Thus, how many undiscovered rework can be detected determines the number of new tasks added in the project scope. The *Scope Growth* in the project is the function of *Rework Discovery*. The value of *Scope Growth* equals *Rework Discovery* multiplies *Ripple Effect Strength*; in which *Ripple Effect Strength* is the project characteristic that describe the strength of ripple effects and it reflects the interdependence of subsystem in the project. The impact of *Ripple Effect Strength* on *Rework Discovery* decides how much new tasks need to be added to complete the project. In my model, I assume the value of *Ripple Effect Strength* is 0.3.

The continuous growth of scope would cause project failure because it diminishes the project progress. In ref. [10], Taylor and Ford demonstrate "Tipping Point" dynamics to control the project scope. A tipping point is a threshold condition that, when crossed, shifts the dominance of the feedback loops that control a process [9]. At the tipping point, the rate of growth project scope and the rate of work being removed from the project backlog are equal. When work is being done faster than new work is being added, the percent of work completed still increases. While work is being done slower than it is being added, the percent of work completed decreases [20]. The system will remain stable as long as conditions remain "below" the tipping point ([9] p.306), but it will become unstable, even lead to project failure when the condition is cross the tipping point.

Once project conditions reach or cross the tipping point, the number of project task backlog needs to be reduced and normally it is implemented through redesign the project. Therefore I built a rate of *Reduce Scope through Redesign* in the model. I use IF THEN ELSE function to formulate this rate. When project conditions reach or cross the tipping point, which means the value of *Scope Growth* bigger than the value of $(Progress\ Rate)_{original}$ to do plus $(Error\ Generation\ Rate)_{original}$ in my model, the *Original Work to do* backlog will reduce the scope through redesign; else no reduction of the project scope. The reduction number is about 8% (This number is based on work quantity reductions presented in ref. [21], Tables 1 and 2.) of the amount of *Original Work to do* backlog. After building this part of the model, I get new equations of *Original Work to do* and *Reduce Scope through Redesign:*

*Original Work to do =Scope Growth-Error Generation Rate*Original Work to do/Total Work to Do- Progress Rate*Original Work to do/Total Work to Do-Reduce Scope through Redesign. The initial value is Initial project size*

*Reduce Scope through Redesign=IF THEN ELSE (Scope Growth>=(Error Generation Rate*Original Work to do/Total Work to Do +Progress Rate*Original Work to do/Total Work to Do), Scope Reduction Fraction*Original Work to do, 0)*

We have to notice that in previous steps of models, because the progress is being done within the project scope, I just formulate *Fraction Actually Done* equals *Work Done* divided by *Initial Project Size.* But in current model, since the progress is being done beyond the project scope, the *Initial Project Size* cannot stand for the total work scope in the project anymore. Therefore, the formulation of project *Fraction Actually Done* changes as a consequence and the same for *Fraction Perceived Done*. In the current model,

Fraction Actually Done= Work Done/Total Project Size
Fraction Perceived Done= (Undiscovered Rework +Work Done)/Total Project Size

In the equations, *Total Project Size* which represents the real whole project scope represented as a stock in the model. Its initial value is *Initial Project Size* and the inflow of it is *Scope Growth* minus *Reduce Scope Through Redesign*.

4.4.3. Haste creates out-of-sequence work

At some point in your life, someone told you that "Haste makes waste" [22]. It normally happened after you had rushed to finish your work and then found that the rework to fix the mistakes would take more time than if you did it carefully at the first pass. During the project, schedule pressure makes employees work faster to catch up the project schedule. Hence, "Haste makes waste" is the secondary impact of schedule pressure. Thibodeau and Dodson suggest that schedule pressure often result in the "overlapping of activities that would have been accomplished better sequentially", and overlapping can significantly increase the chance of errors [23]. For example, in a software development project, the coding work should start after all the design works are fully completed, because a correct design is the essential basis of the further coding work. If employees are hurry to start coding work before the design works are entirely finished due to the schedule pressure, this overlapping work condition could make errors significantly increasing.

In my model, the growth of the *Original Work to do* leads to the increase of *Total Work to do* and *Known Work Remaining*, so the *Schedule Pressure* increases as a consequence. Moreover, the increasing Schedule Pressure affects the Error Fraction. Therefore, the increasing *Original Work to do* does indirectly result in the increase of *Error Fraction* through *Schedule Pressure*. Figure 43 shows those relations in the model.

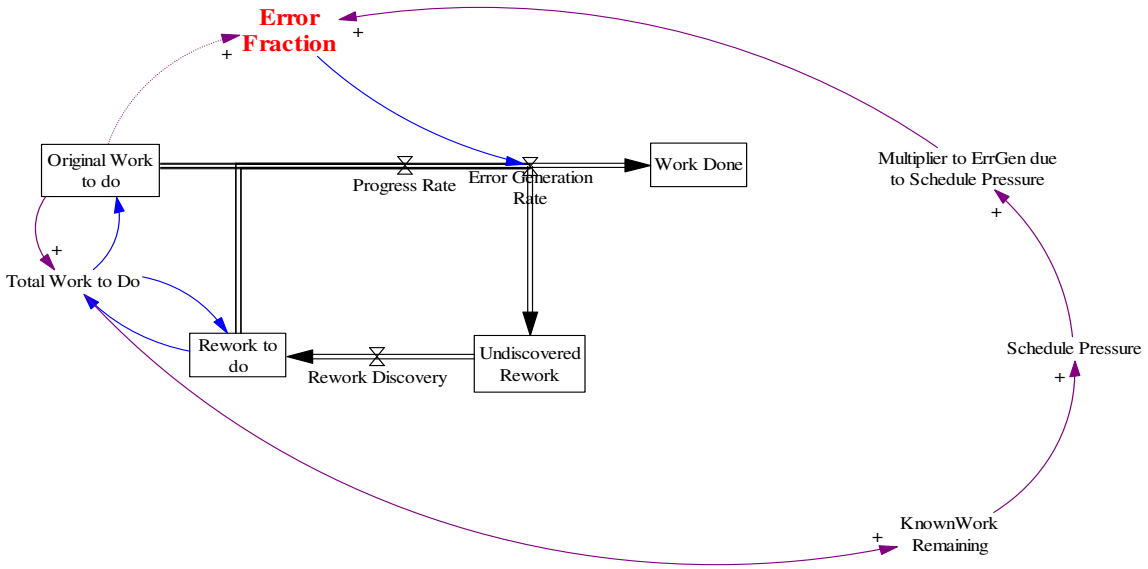


Figure 43 Haste creates out-of-sequence work. ------> indicates the overlap and concurrence knock-on effect shown in the causal loop diagram in ref. [5] , ——> indicates how this effect is implemented in my model.

In the section 4.3.2.4 I have discussed how *Schedule Pressure* affects *Error Fraction*. I introduce a *Multiplier to error generation due to schedule pressure* to formulate the effects which are caused by *Schedule Pressure*. Since overlap and out-of-sequence work is the secondary effect of *Schedule Pressure*, I don't introduce another multiplier to formulate the effect to error fraction due to overlapping work in this step. The multiplier to error fraction due to "Haste creates out-of-sequence work" effect is included in *Multiplier to error generation due to schedule pressure*.

4.4.4. Hopelessness

Working overtime and adding pressure to staff in order to increase the working rate are common ways to avoid disruption on delivery. Although these actions used to increase the working productivity levels, effects on morale would actually lead a lower level of productivity and higher level of error generation via fatigue and increase of rework. Therefore, impacts caused by the changes of morale are knock-on effects of policy resistance to improve schedule performance and those impacts can create a sense of "hopelessness" in the process of project.

4.4.4.1. Building morale

In the ref. [5], they demonstrate that *Morale* is affected by *Rework to do* and Fatigue in the causal loop diagram. In my model, I build *Morale* as the long term effect of sustained pressure. Figure 44 shows the causal loop diagram of building morale in my model.

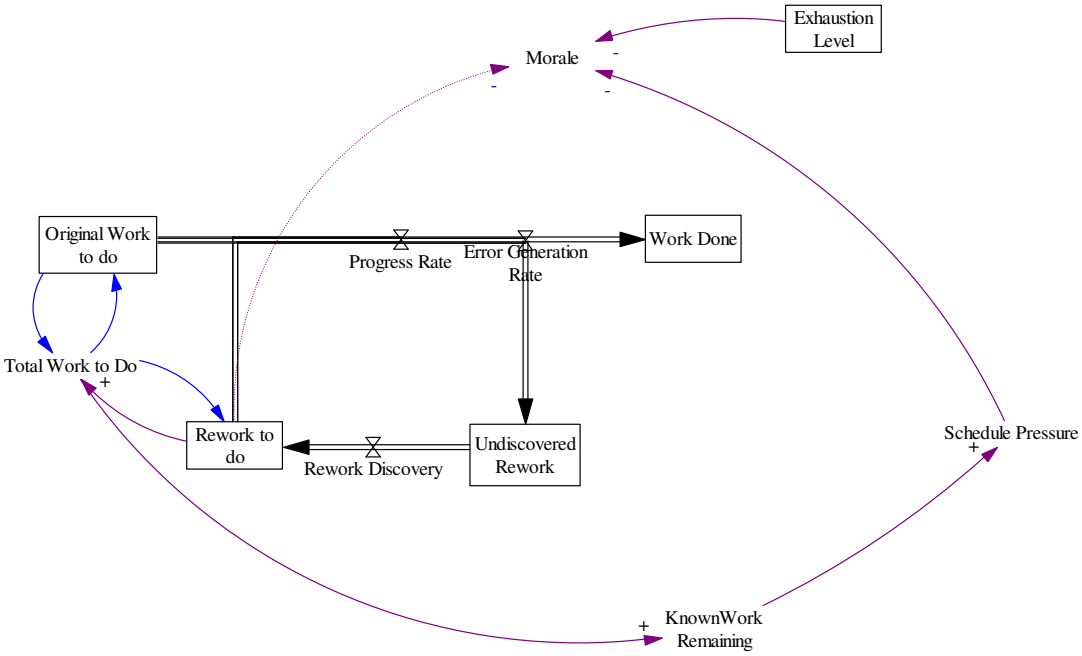


Figure 44: Diagram of building morale. ------► indicates the overlap and concurrence knock-on effect shown in the causal loop diagram in ref. [5] paper, ——► indicates how this effect is implemented in my model.

The same as the growth of *Original Work to do*, the increase of the *Rework to do* also leads the increase of *Total Work to Do* and *Known Work Remaining*, so the *Schedule Pressure* increases as consequence and a high level of schedule pressure could cause a low level of morale. Thus, the connection between *Rework to do* and *Morale* in the ref. [5]'s diagram is implemented by the impacts on *Morale* due to *Schedule Pressure* in my model shown in Figure 44. In addition, the increase of exhaustion level decreases the morale too.

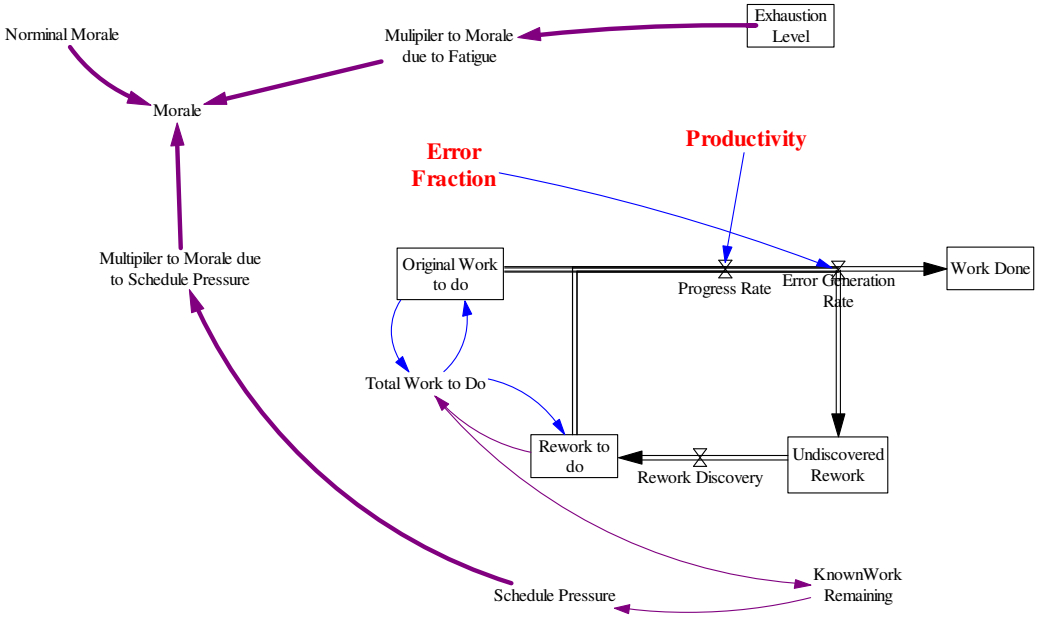


Figure 45: Building Morale

Figure 45 illustrates the model of building morale part. I introduce a parameter *Nominal Morale* which stands for the level of employees' morale in normal working condition (without overwork and schedule pressure) in the model. In order to formulate the impacts to *Morale* due to Fatigue and *Schedule Pressure*, I assume two look-up tables as the multipliers ——*Multiplier to Morale due to Fatigue* and *Multiplier to Morale due to Schedule Pressure.*

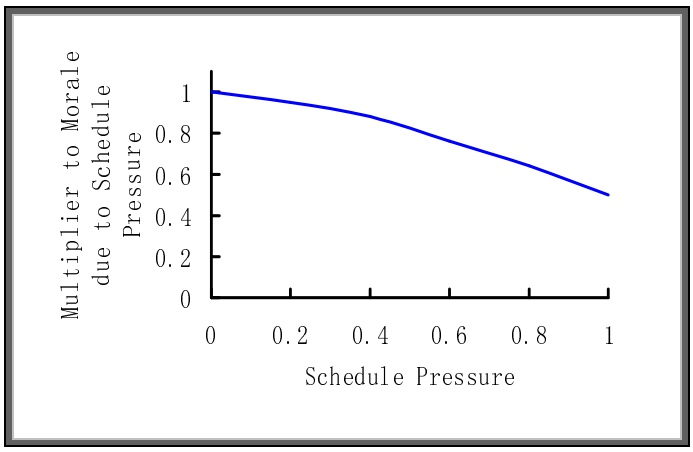


Figure 46: Multiplier to Morale due to Schedule Pressure

Figure 46 shows the lookup table I assumed for *Multiplier to Morale due to Schedule Pressure.* When no *Schedule Pressure* in the project, *Morale* keeps its nominal value and the multiplier is 1. As the *Schedule Pressure* increases, the multiplier decreases. The curve decreases slowly in low *Schedule Pressure* area because people could tolerate some level of pressure, but when the pressure keeps getting bigger, the value of morale would drop faster and faster. The value of *Morale* can reduce to half of its nominal value at the point that *Schedule Pressure* reaches the maximum 1.

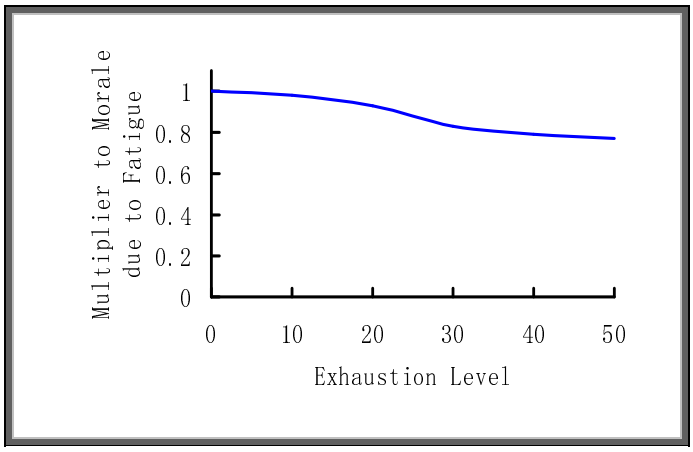


Figure 47: Multiplier to Morale due to Fatigue

Figure 47 shows the lookup table I assumed for *Multiplier to Morale due to Fatigue*. When

exhaustion level is 0, which means people working in the nominal condition without fatigue, the multiplier is 1 and *Morale* is not affected. As the exhaustion level increases, the multiplier decreases. We have to notice that in the causal loop diagram in ref. [5], *Fatigue* is caused by *Overtime*, so *Fatigue* can just affects *Morale* when people are working overtime. That means *Multiplier to Morale due to Fatigue* is used to formulate *Morale* when *AFMDP* is bigger than 1. I use IF THEN ELSE function to implement it. The equation of Morale is my model is

*Morale= IF THEN ELSE (Actual Fraction of A Man Day on Project>1, Nominal Morale*Multiplier to Morale due to Fatigue*Multiplier to Morale due to Schedule Pressure, Nominal Morale*Multiplier to Morale due to Schedule Pressure)*

4.4.4.2. Effects due to Morale

As I mentioned before, morale problems can create a sense of "hopelessness" that increases error generation and decreases productivity, and which also increases turnover. Figure 48 illustrates the structure of "hopelessness" part in the model.

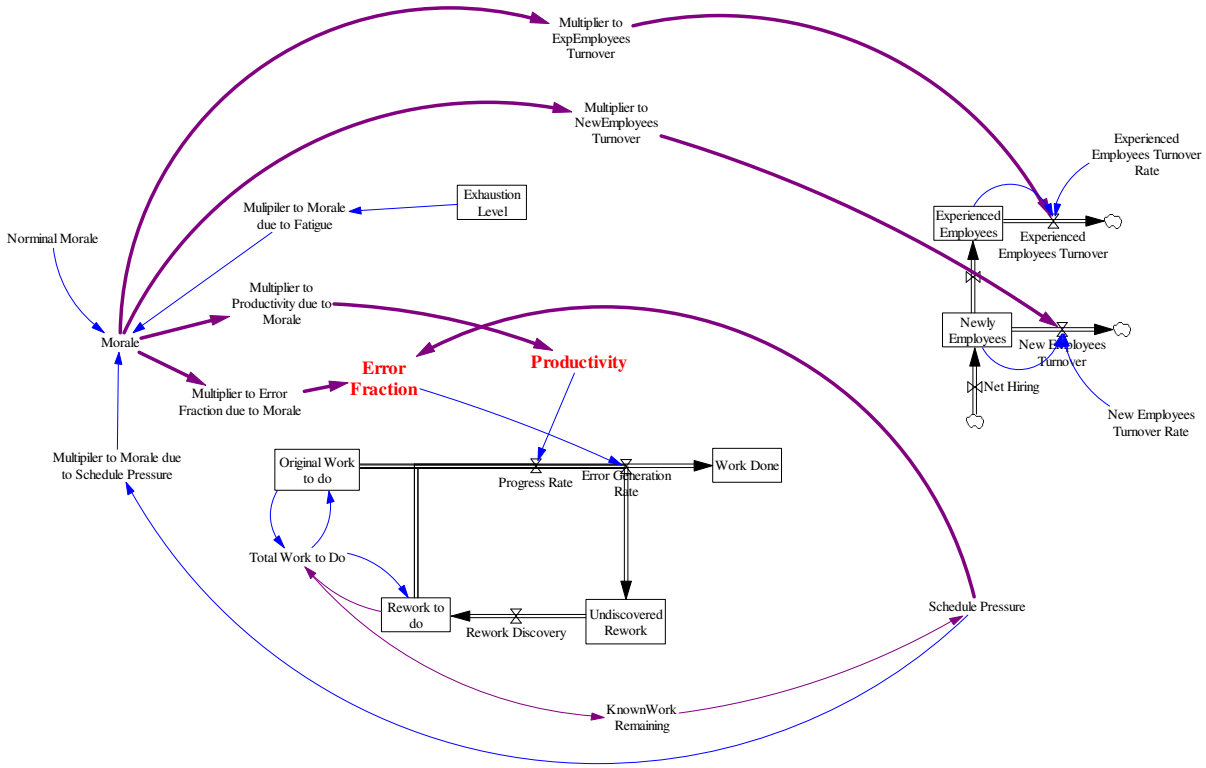


Figure 48: Hopelessness

The same as many other parts in my model, I still using assumed multipliers to formulate the effects to error generation, productivity and turnover. Figure 49 shows the lookup graph I assumed for *Multiplier to Productivity due to Morale.*

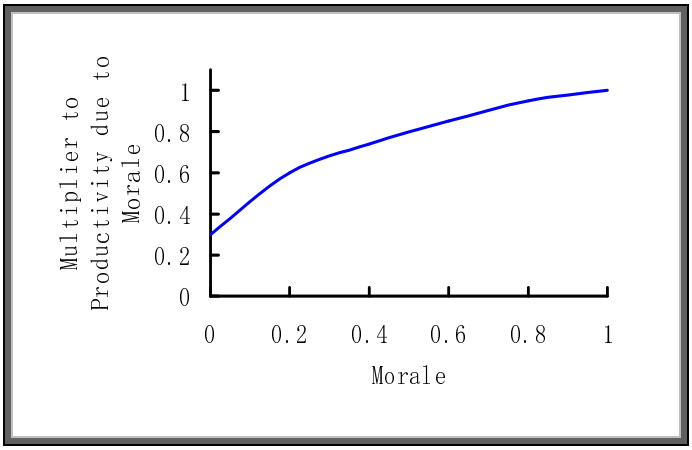


Figure 49: Multiplier to Productivity due to Morale

Because 1 is the nominal value of morale, productivity is not affected by morale in this condition. The multiplier is 1. When morale decreases, the productivity reduces as follow. Imagine if people's morale decreases from 1 to 0.8, even though it affects productivity, the value of productivity won't decrease very much because people can still stand this level of morale. But if employees' morale decreases from 0.4 to 0.2, the productivity will drop very much because the sense of "hopelessness". People are already very tired or being in high pressure condition, they don't want to handle more. That is why the curve drops faster in low morale area than high morale area and the productivity could drop 70 percent if the value of morale is 0.

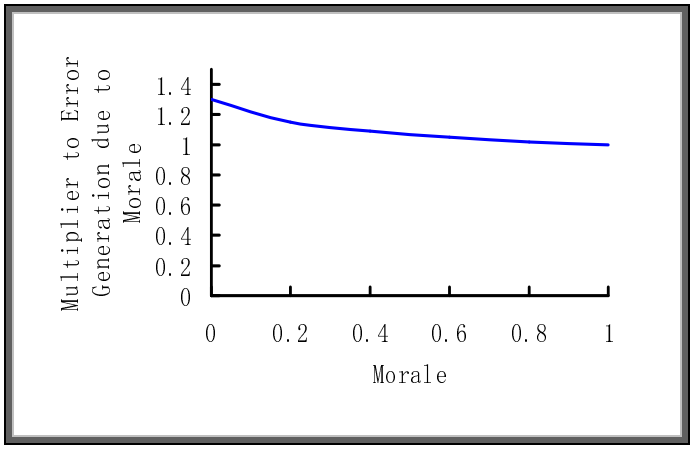


Figure 50: Multiplier to Error Fraction due to Morale

Figure 50 illustrates the lookup table of *Multiplier to Error Fraction due to Morale*. When the value of morale is 1, the multiplier is 1. Decreasing value of morale increases the error generation, because when people are tired and in a high schedule pressure, they have difficulties to concentrate in the quality. From the figure, we can see that in my lookup graph *Morale* doesn't affect error generation very much. That is because the effect on *Morale* is the secondary effect of *Schedule Pressure* and in my model there have already has a *Multiplier to Error Generation due to Schedule Pressure*. In addition, when employees are in a low spirits,

they intend to work less instead of making more mistakes, so the main effect of *Morale* is on *Productivity* in real projects. Thus, the *Multiplier to Error Fraction due to Morale* is just up to 1.3 in the graph.

After the analysis above, we get the updated equations of *Productivity*, *Error Fraction* in my model.

Productivity= IF THEN ELSE (Actual Fraction of A Man Day on Project<=1, Actual Fraction of A Man Day on Project(1-Congestion and Communication Difficulties)*Fraction Satisfactory*Potential Productivity*Multiplier to Productivity due to Morale, Fraction Satisfactory * Potential Productivity *(1-Congestion and Communication Difficulties)*Multiplier to Productivity due to Morale)*

*Error Fraction= IF THEN ELSE(Actual Fraction of A Man Day on Project>1, Nominal Error Fraction*Multiplier to ErrGen due to WF Mix*Multiplier to ErrGen due to Schedule Pressure*Multiplier to ErrGen due to Fatigue*Multiplier to Active Error Regeneration Due to Error Density*Multiplier to Error Fraction due to Morale, Nominal Error Fraction*Multiplier to ErrGen due to WF Mix*Multiplier to ErrGen due to Schedule Pressure*Multiplier to Active Error Regeneration Due to Error Density*Multiplier to Error Fraction due to Morale)*

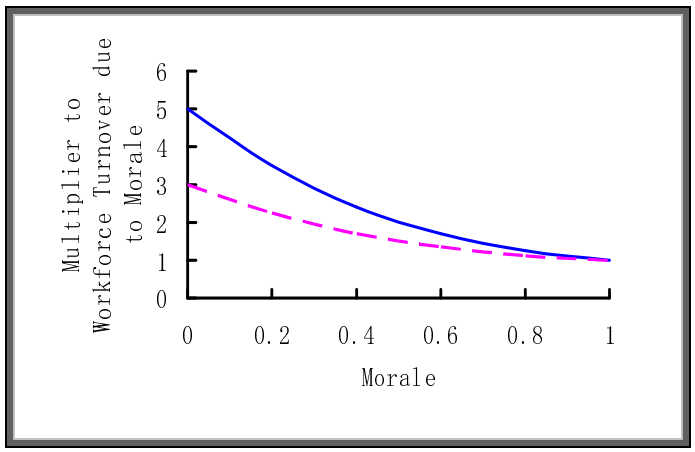


Figure 51: Multipier to Workforce Turnover due to Morale

In previous steps, I did not consider about the turnover of workforce. In real life project, the turnover must exist, so I add this part in this step. Since the experienced employees are more productive and can handle the tasks better than new employees, the experienced employees' turnover rate is less than new employees. I assume that *Experienced Employees Turnover Fraction* is 0.0001 (1/day) and *New Employees Turnover Fraction* is 0.0002 (1/day). Morale

affects workforce turnover, the lookup graph of *Multiplier to Workforce Turnover due to Morale* shown in Figure 51. As the same as last two lookup graph, when *Morale* is 1, the multiplier to workforce turnover is 1. The multiplier increases follow the decrease of *Morale*. The curve increases very fast in low morale area because stronger "hopelessness" sense causes more workforce turnover. It is easy to imagine that experienced workforce can tolerate more "hopelessness" sense than new employees, so the multiplier of experienced employees' turnover should less that new employee's turnover. In the graph the actual line stands for the *Multiplier to New Workforce Turnover due to Morale*, the dotted line stands for *Multiplier to Experienced Workforce Turnover due to Morale*. From the figure I can see that *Multiplier to New Workforce Turnover due to Morale* can be up to 5, while the maximum value of *Multiplier to Experienced Workforce Turnover due to Morale* is just 3. The equations of turnover rates in the model are:

*Experienced Employees Turnover= Experienced Employees Turnover Fraction*Experienced Workforce*Multiplier to ExpWorkforce Turnover due to Morale*

*Newly Employees Turnover= New Workforce*Newly Employees Turnover Fraction *Multiplier to NewWorkforce Turnover due to Morale*

4.4.5. Model Validation and behavior comparison

I test the model incrementally by adding the knock-on effects one by one. Recall that I built the knock-on effect "Errors build Errors" first in this step. To implement this, I divided *Undiscovered Rework* level into two levels: *Undiscovered Active Rework* and *Undiscovered Passive Rework*. Upstream undiscovered active errors can create at least one downstream error. There is an auxiliary named *Active Errors Fraction* in the model to decide the proportion of active errors in *Error Generation Rate*. Thus, if I set 0 to the value of *Active Errors Fraction,* which means there is no active error in the project, *Active Error Fraction in Task Perceived Done* should be 0 and no further error due to upstream undiscovered active errors would be generated. In this condition, *Multiplier to Error Generation due to Active Error Regeneration* is always 1 and it would not impact the error fraction at all. Therefore, the simulation result of this model should be totally the same as the model of previous step.

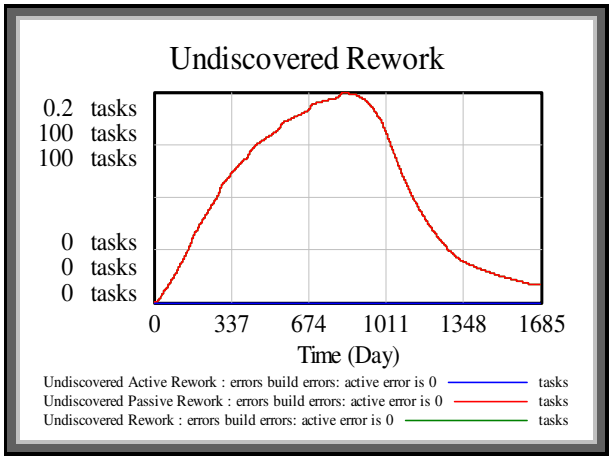


Figure 52: Errors build errors: Undiscovered Reworks in the condition of active error is 0

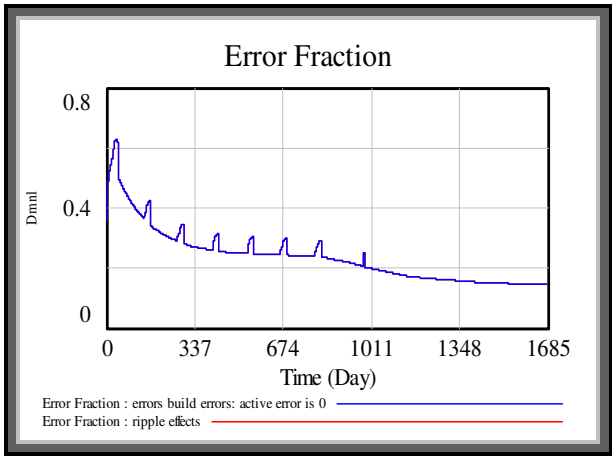


Figure 53: Error Fraction in ripple effects step and errors build errors step (active error is 0)

Figure 52 and 53 show the simulation result when *Active Error Fraction* is 0. From Figure 52 we can see that the *Undiscovered Active Rework* is 0 all the time, *Undiscovered Passive Rework* equals total *Undiscovered Rework*. In Figure 53, the graph of *Error Fraction* in step 4 —adding ripple effects step overlaps the graph of *Error Fraction* in "error build errors" step when *Active Error Fraction* is 0.

After considering the condition *Active Errors Fraction* is 0, I start to discuss another limit— *Active Errors Fraction* is 1 next. When *Active Errors Fraction* is 1, no undiscovered passive error exists in the project at the begging, total *Undiscovered Rework* equals *Undiscovered Active Rework*. But as the process goes on, because of the *Active Error Retirement Rate*, *Undiscovered Passive Rework* starts to show and the total *Undiscovered Rework* becomes the sum of *Undiscovered Active Rework* and *Undiscovered Passive Rework*, shown in Figure 54. Since undetected active errors exist in the model, they will regenerate more downstream errors. Undiscovered active errors affect *Error Fraction* through *Multiplier to Error Generation due to Active Error Regeneration*. Thus the value of *Error Fraction* should be bigger than the value in ripple effects step, especially in the beginning of the project because all errors are active ones then, as shown in Figure 55.

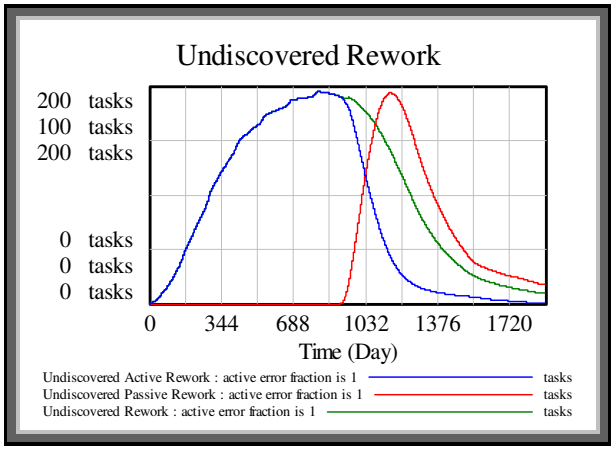


Figure 54: Errors build errors: Undiscovered Reworks in the condition of active error fraction is 1

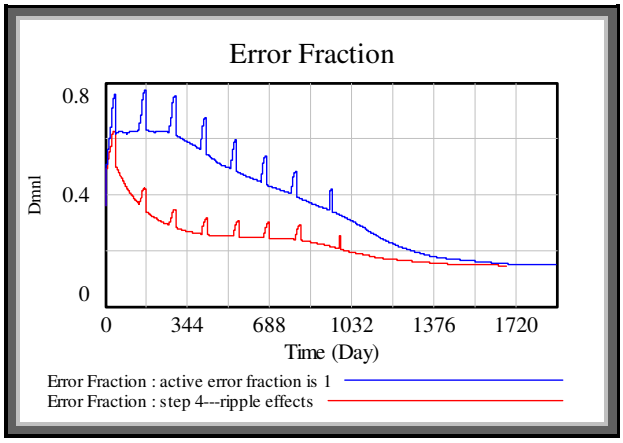


Figure 55: Error Fraction in ripple effects step and errors build errors step (active error fraction is 1)

All graphs I gave match my analysis above, so I believe that my model is correct after adding the "errors build errors" effect.

The second knock-on effect I added to the model was "errors create more work". The process of fixing discovered errors would cause new tasks that create development activities beyond the previous project scope.

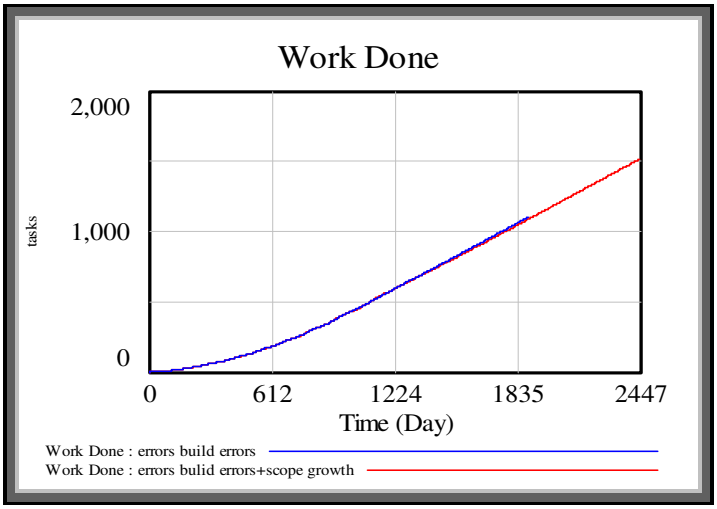


Figure 56: Works beyond the scope

From Figure 56 we can see that after adding effect of "errors create more work" in the model, the total number of work done in the project is around 1500, while the amount is only the initial project size 1200 before adding this effect. Thus, the current scope goes beyond the initial project scope. In addition, because the total project scope increases, it takes longer time to finish the project as a consequence.

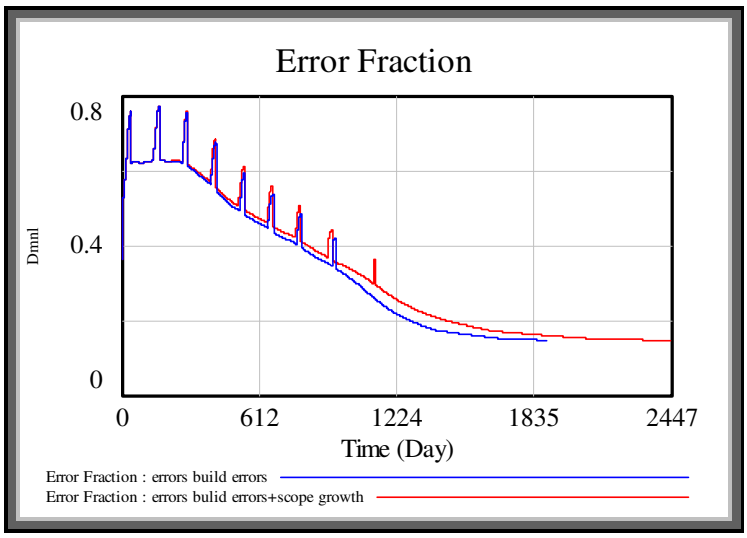


Figure 57: Error Fraction with scope growth Vs. Error Fraction without scope growth

The growth of the scope does not affect error generation directly, so the graph of *Error Generation* should not change very much comparing with the graph before adding effect of "errors create more work", shown in Figure 57. Because total project size increases during the project, the speed of *Fraction Actually Done* decreases and the *Nominal Error Fraction* increases as consequence. That is why *Error Fraction* after adding effect of "errors create more work" is bigger than before.

In the section 4.4.3 I have discussed that the effect of "Haste creates out-of-sequence work" results the increase of Error Fraction through *Schedule Pressure*. Therefore, I don't add any variable in this step to implement it.

The last knock-on effect I added in the model was effects on *Morale* and I call the model "final model" after adding this part. Because I just use multipliers to implement it, it would not affect model structure. Thus, the way of model testing I choose here is behaviors comparison, to see if the current behaviors make sense.

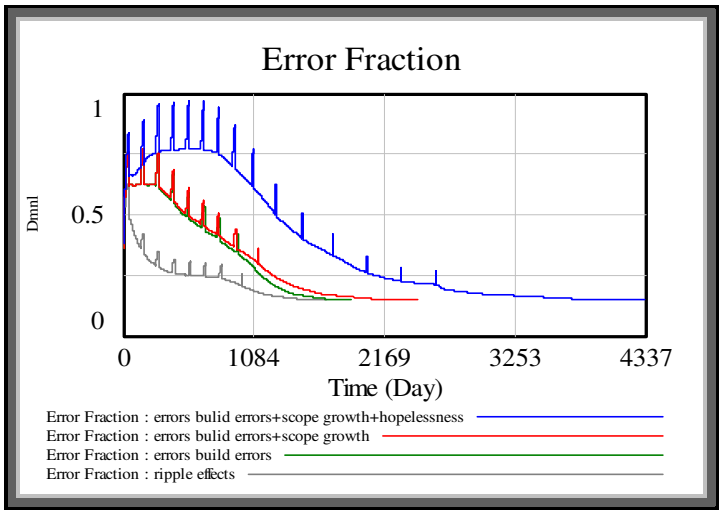


Figure 58: Error Fraction in different models

Figure 58 shows *Error Fraction* in different steps of models. Even though in my model the *Multiplier to Error Fraction due to Morale* is pretty small, but a small increase of error generation rate can lead to a big multiplier due to error regeneration because every undetected active error will lead to more than one error downstream. And this big multiplier due to error regeneration will affect back to error fraction. That is why the graph after adding effect of morale is much bigger than the graph in previous step in the figure. Moreover, we can see that the boost in the graph of the final model is higher than other steps. That is because fatigue just affects morale when *AFMDP* is bigger than 1 and the boost just happens in working overtime period.

Effects on morale increase the error fraction and decrease the productivity. In addition, more error generation causes high rework discovery rate and then leads to more scope growth. Therefore, after adding the effect on morale, the deadline of the project will be slipped, more employees are needed. The simulation results are shown in Figure 59 and 60.

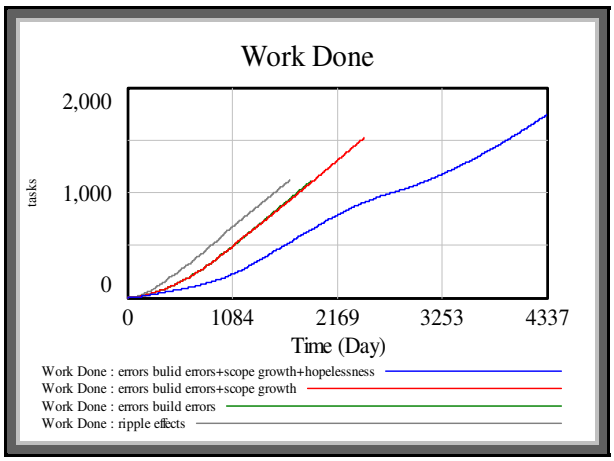


Figure 59: Work Done in different models

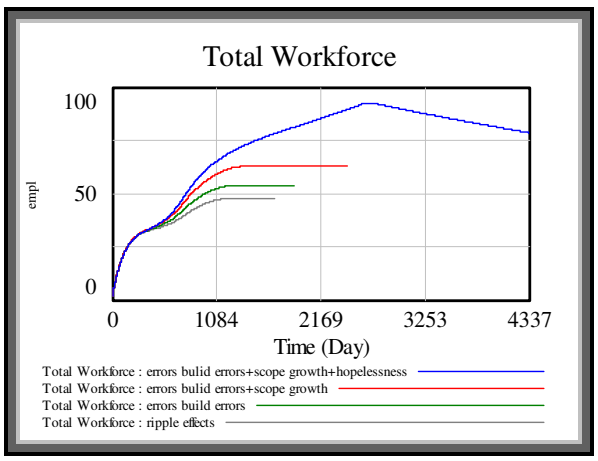


Figure 60: Total Workforce in different models

We notice that the value of total workforce in final model suddenly drops in the days around 2500. That is because project scope reduces through redesign. The simulation results above are all making sense. After model validation, I believe that my final model is correct.
This concludes the final version of my model.

5. Policy Analysis and Recommendations

Most research and applications in project dynamics has focused on explaining the reasons of cost and schedule overrun in particular situations and developing actions that could reduce the overruns. How could the project deviate from their original plan, and why? Post-projects assessment of real cases shows that most big cases often involve disputes between the owner of the project and the executer of the project. For example, the famous case between Ingalls Shipbuilding and the U.S. Navy described in ref. [3]. In the project, owners often request changes to the planned contract, such as adding specific requirements, increasing the project scope and modifying the original design. Those disputes can cause delay and disruption in many cases. In addition to the changes to the plan, another common trigger to an unexpected project result is underestimating work scope. Projects that are underestimated would end up costing more because a too aggressive project plan leads to high schedule pressure. This high pressure would cause make-up actions like working overtime and later ripple and knock-on effects that actually make the performance of the project worse later.

Projects rarely go as planned. What should the project managers do to avoid overruns? I am going to give some recommendations below according to project dynamics learning and the simulation results of my model.

(1) Working with a fixed the contract: As I mentioned above, disputes between project owner and executer are improved to be one of the most important reasons of causing the project failure by many real world cases. Therefore, in order to avoid disputes during the project, the owner of the project must decide a fixed project scope and present the full requirements to the executer before signing the contract. Once signing, this contract should not be changed anymore during the process of project.

(2) Making a reasonable project plan: A project plan sow the seeds for a projects success or failure. Projects that both are underestimated and overestimated cost more in the end than they are thought. Thus we have to develop project plans more reasonable and realistic before starting the work. This project plan includes desired project budget, deadline and work force resources. Take my model as an example.

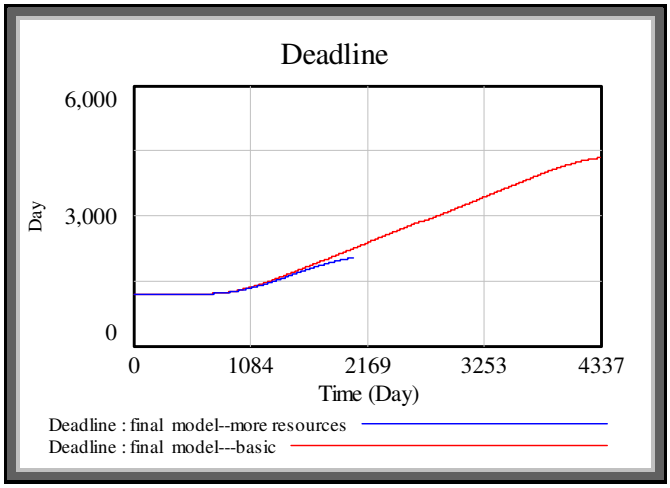


Figure 61: Project progress comparisons between basic final model and adding more resources model

In my model, the initial work force number is 2 and the project takes more than 4000 days to end up with this initial amount of work force. However, when I increase the initial number of work force up to 20 and simulate again, the whole project just uses around 2000 days to finish, much less than the value of the basic final model, shown in Figure 61. From the simulation results, I can conclude that in the basic final model, the project plan of initial work force is probably underestimated. When project managers make the project plan they ignore the factors that in real life nobody can work without making mistakes and some slack time is required while working. Those being ignored factors cause the original project plan unrealistic so that result a big project deadline extension.

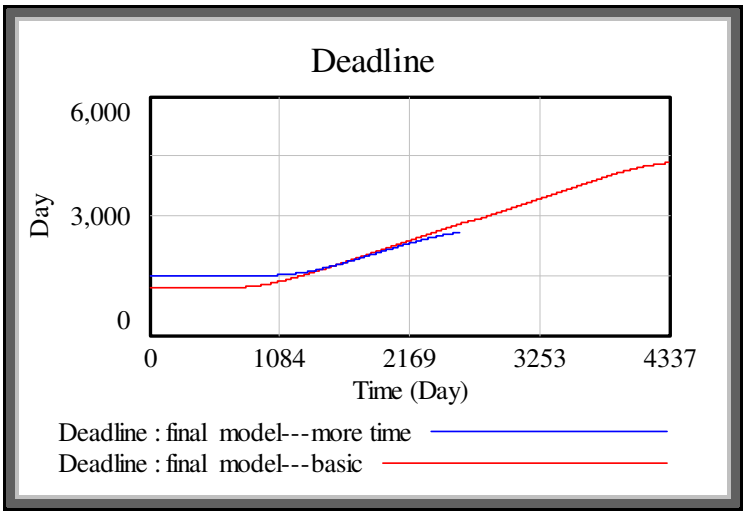


Figure 62: Project progress comparisons between basic final model and adding more time model

The same as work force, the completion date is probably underestimated too due to the simulation result. In my model, the initial project completion date is the 1200th day, but the final result is far more than this planned date. When I increase value of *Initial Deadline* from 1200 to 1500, the simulation result is much better, shown in Figure 62. A

too aggressive project plan, no matter in the short of work force or project completion date, would result schedule pressure and increase of work intensity even working overtime. And almost all the ripple and knock-on effects are the primary or secondary effects of schedule pressure and working overtime. Therefore, the first step to avoid adverse project dynamics is to bid and plan the project correctly and my model improve it too.

(3) Managing the rework cycle: Besides taking those preventions I give above before the project starting, what should project managers respond when problems occur? Rework cycle is the central factor to many adverse project dynamics. A full recognition of rework cycle can help project managers to minimize the bad consequences. Try to slow down the working speed and do work right at the first pass even at a low productivity. *Undiscovered Rework* is the most important feature in rework cycle. The undetected active rework can lead a big multiplier to error fraction due to "errors build errors" effects. Therefore, detecting undiscovered rework is badly needed. In addition, avoid the tendency to start downstream work too early to generate working overlap.

(4) Reducing work force *Average Assimilation Delay*: New employees are productive less than experienced ones in the project, so reinforcing the training intensity to new employees to make them become experienced ones will increase the whole project productivity and improve the project progress. The comparison of simulation results are shown in the Figure 63. In the basic final model the average assimilation delay is 80 days and I decrease it to 50 days in the "less WF assimilation" model. From the simulation results we can see that reducing work force average assimilation delay is a good way to improve the project progress.

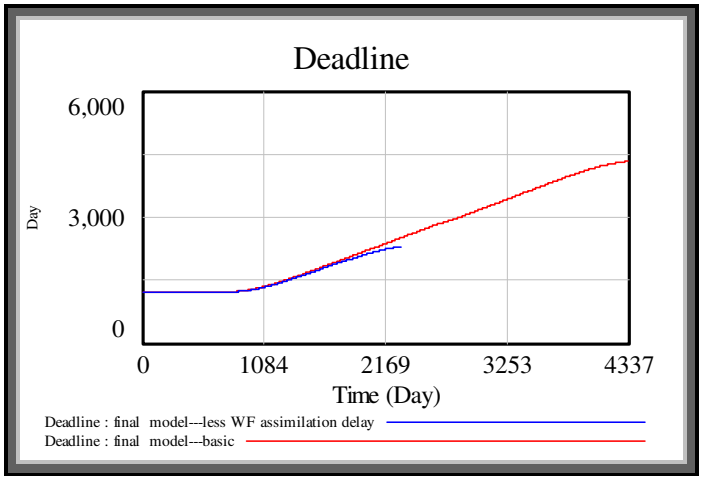


Figure 63: Project process comparisons between basic final model and less WF assimilation delay model

(5) Morale encouragement: Morale is another essential factor that impacts project progress. It increases error generation, increases workforce turnover rate and decreases productivity. Thus, when employees are in a very low morale, project managers should use means to encourage employees to pull their morale level back.

6. Discussion and Conclusion

The model I create in this master project is a generic project dynamic model that includes all primary causes of project dynamics—project features, the rework cycle, project controls, and ripple and knock-on effects. At the beginning the project I assumed for my model plans to use 1200 days to finish works, but finally it takes more than 4000 days before it ends up, far more their anticipation. We can get the reason why this result happens through analyzing the model. Project mangers ignore realistic factors (e.g. everyone makes mistakes; people need slack time while working) so that they make an aggressive project plan. This plan leads a schedule pressure as the project is going on and in order to catch up with the schedule, employees choose to increase working intensity or even working overtime. However, the unintended and undesired impacts on worker productivity and work quality (e.g. ripple and knock-on effects) which are generated by those projects control policies (e.g. increasing working intensity and working overtime) make the project progress even worse.

From my work, we can see that system dynamics model can be used to quantify and explain the impact of these direct changes to the project on its final cost and it is also suited to determine the magnitude of these unintended ripple and knock-on effects and explain their origins. Therefore, system dynamic models should be often used for risk assessment of projects. There are two ways for system dynamic model use for risk assessment:

(1) Post-project assessment: System dynamics is a scientific method for assessing what went right and what went wrong in a project. Learning how the project deviated from the original plan and from previous unsuccessful cases is a very important guide for what might occur in future projects, so that project managers could do something to avoid such failure situations to happen again. Documentation of success stories and assessment process can facilitate this progression.
(2) Pre-project simulation: In the previous chapter I have discussed what an important role the project estimation plays in a project. How could project managers make a more accurate project plan? How could they know how strong the ripple and knock-on effects are in the project? Besides using prior project experiences, a pre-project simulation can also be used to develop base estimates. Managers could create a simple dynamic model according to their estimates of the project before starting the work, and see what is going to happen in further process of the project in those estimates through simulation results.

System dynamics has proven to be an effective methodology to explain the reasons of project failure. Although projects from different industries have their own particular characteristics, still some similarities exist. From my work of building this generic system model, I would expect that system dynamics models are commonly used to improve project management.

References

[1] Stephens CA, Graham AK, Lyneis JM. 2005. System dynamics modeling in the legal arena: meeting the challenges of expert witness admissibility. *System Dynamics Review* 21(2): 95 – 122.

[2] Ackermann F, Eden C, Williams I. 1997. Modeling for litigation: mixing qualitative and quantitative approaches. *Interfaces* 27(2): 48-65.

[3] Cooper KG. 1980. Naval ship production: a claim settled and a framework built. *Interfaces* 10(6): 20 – 36.

[4] Project failures in IT industry http://www.it-cortex.com/Examples_f.htm

[5] Lyneis JM, Ford DN. 2007. System dynamics applied to project management: a survey, assessment and directions for future research. *System Dynamics Review* 23 (2/3):157-189.

[6] System dynamics, http://en.wikipedia.org/wiki/System_dynamic

[7] Levitt RE, Thomsen J, Christiansen TR, Kunz JC, Jin Y, Nass C. 1999. Simulating project work processes and organizations: toward a micro-contingency theory of organizational design. *Management Science* 45(11): 1479 – 1495.

[8] Cooper KG. 1993. The rework cycle (a series of 3 articles): why projects are mismanaged; how it really works . . . and reworks . . . ; benchmarks for the project manager. *PMNETwork* February (for first two articles); *Project Management Journal* March (for third article).

[9] Sterman JD. 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Irwin/McGraw Hill: Chicago, IL.

[10] Taylor T, Ford DN. 2006. Tipping point dynamics in development projects. *System Dynamics Review* 22(1): 51 – 71.

[11] system methodology http://sysdyn.clexchange.org/sd-intro/home.html

[12] Abdel-Hamid TK, Madnick SE. 1991. *Software Project Dynamics: An Integrated Approach*. Prentice-Hall: Englewood Cliffs, NJ.

[13] McGowan CL. Management Planning for Large Software Projects. *IEEE*, 1978.

[14] Brooks FP. Jr. *The Mythical Man-Month*. Reading, MA: Addison-Wiley Publishing Co. 1978.

[15] Deutsch MS. *Verification and Validation*. Software Engineering. Edited by R.W. Jensen and C. C. Tonies. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1979.

[16] Albert DS. The Economics of Software Quality Assurance. *National Computer Conference*, 1976.

[17] Belford PC et al. An Evaluation of the Effectiveness of Software Engineering Techniques. *IEEE COMPCON, (Fall 1977).*

[18] Mills HD. *Software Productivity*. Canada: Little, Brown &Co., 1983.

[19] Jones TC. Defect Removal: A Look at the State of the Art. *ITT CommNet,* Vol.1, No.3 (December 1981).

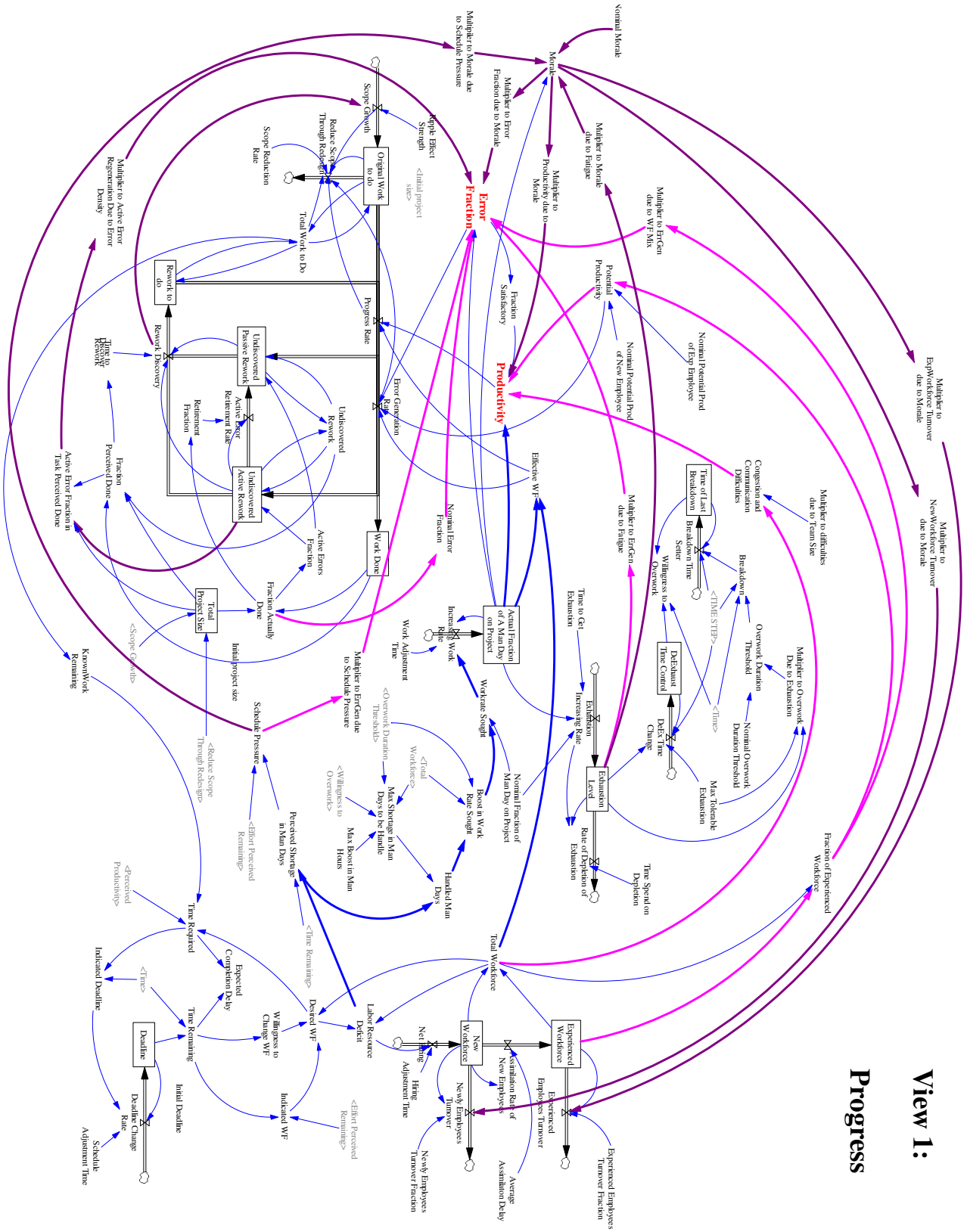
[20] Taylor T, Ford DN. Managing Tipping Point Dynamics in Complex Construction Projects. *ASCE Journal of Construction Engineering and Management.* Vol. 134, No. 6, pp. 421-431. June, 2008.

[21] Clarey J. 1987. New concept cuts nuclear project construction cost. *Power Eng.* J., 91-11, 28–31.

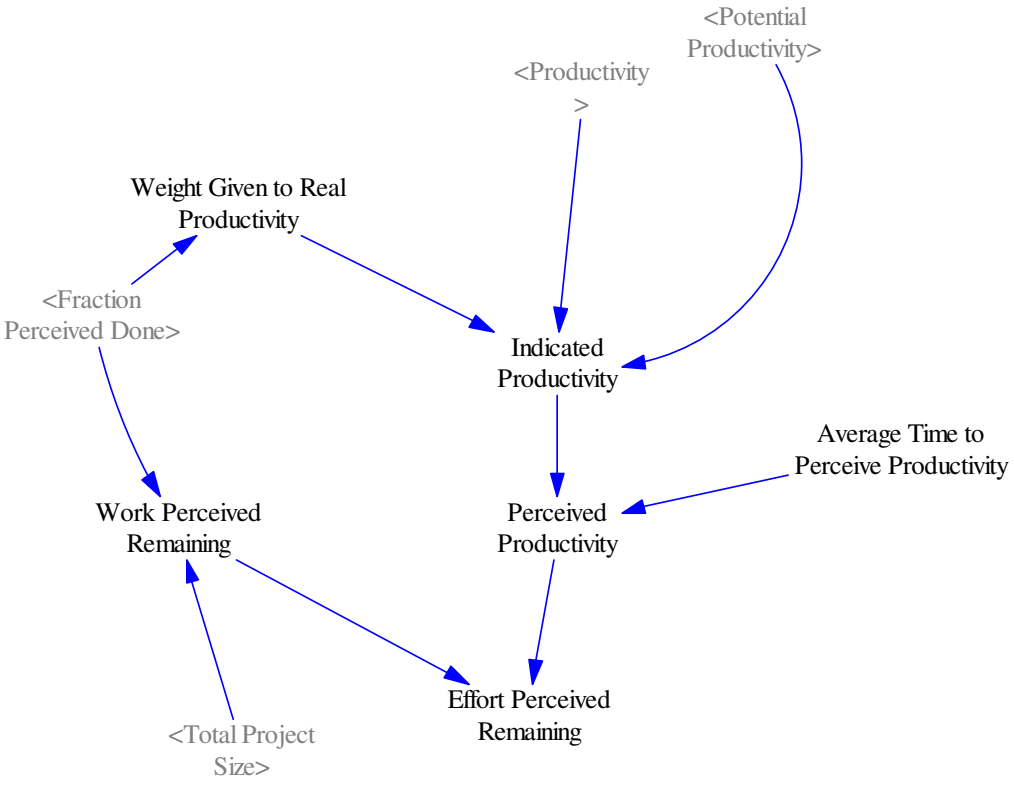
[22] Cooper KG. 1994. The $2,000 hour: how managers influence project performance through the rework cycle. *Project Management Journal* 25(1).

[23] Thibodeau, R. and Dodson, E.N., Life Cycle Phase Interrelationships. *Journal of Systems and Software*, Vol.1, 1980, 203-211

Appendix A – Full SD-models and equations



View 2: Effort.



Equations:

Stocks:

Actual Fraction of A Man Day on Project=

INTEG (Increasing Work Rate, Nominal Fraction of Man Day on Project)

Units: 1/Day

Actual fraction of a man-day on project. The slack time is the fraction of project time lost to non-project activities, such as coffee and breaks. 100% increase is attainable because workers. In addition to partially compressing their slack time, may also work overtime hours. This will cause actual productivity to be larger than potential productivity. The motivational effects of schedule pressure can push AFMDP to higher values.

Deadline=

INTEG (+Deadline Change Rate, Initial Deadline)

Units: Day

DeExhaust Time Control=

INTEG (DeEx Time Change,0)

Units: Day

Exhaustion Level=
INTEG (Exhaustion Increasing Rate-Rate of Depletion of Exhaustion,0)
Units: Dmnl
Exhaustion is defined in the model due to overwork

Experienced Workforce=
INTEG (Assimilation Rate of New Employees-Experienced Employees Turnover,2)
Units: empl
The number of experienced employees working in the project

New Workforce=
INTEG (Net Hiring-Assimilation Rate of New Employees-Newly Employees Turnover,0)
Units: empl
The number of new employees working in the project

Original Work to do=
INTEG (Scope Growth-Error Generation Rate*Original Work to do/Total Work to Do- Progress Rate*Original Work to do/Total Work to Do-Reduce Scope Through Redesign, Initial project size)
Units: tasks
The number of original unsuccessfully solved tasks

Rework to do=
INTEG (Rework Discovery-Error Generation Rate*Rework to do/Total Work to Do-Progress Rate *Rework to do/Total Work to Do, 0)
Units: tasks
The number of unsolved rework

Time of Last Breakdown=
INTEG (Breakdown Time Setter,-1)
Units: Day

Total Project Size=
INTEG (+Scope Growth-Reduce Scope Through Redesign,Initial project size)
Units: tasks

Undiscovered Active Rework=
INTEG (Error Generation Rate*Active Errors Fraction-Active Error Retirement Rate-Rework Discovery*Undiscovered Active Rework/(Undiscovered Rework+0.001), 0)
Units: tasks
Parts of undiscovered rework which can regenerate further errors

Undiscovered Passive Rework=
INTEG (Active Error Retirement Rate+Error Generation Rate*(1-Active Errors Fraction) -Rework Discovery*Undiscovered Passive Rework/(Undiscovered Rework+0.001), 0)

Units: tasks
Parts of undiscovered rework which cannot regenerate further errors

Work Done=
INTEG (Progress Rate, 0)
Units: tasks
The number of tasks successfully processed

Flows:

Active Error Retirement Rate=
Undiscovered Active Rework*Retirement Fraction
Units: tasks/Day
When undiscovered active errors cease to reproduce, they effectively become undiscovered passive errors. The active error retirement rate is regulated through the fraction of active errors that become passive every unit of time.

Assimilation Rate of New Employees=
New Workforce/Average Assimilaton Delay
Units: empl/Day
The rate at which people go from being new to becoming experienced in the project.

Breakdown Time Setter=
(MAX (Time of Last Breakdown, Breakdown)-Time of Last Breakdown)/TIME STEP
Units: Dmnl

Deadline Change Rate=
(Indicated Deadline-Deadline)/Schedule Adjustment Time
Units: Dmnl

DeEx Time Change=
IF THEN ELSE (Exhaustion Level/Max Tolerable Exhaustion>=0.1, 1, -DeExhaust Time Control
/TIME STEP)
Units: Dmnl
De-Exhaustion time change rate

Exhaustion Increasing Rate=
WITH LOOKUP (((1-Actual Fraction of A Man Day on Project)/ (1-Nominal Fraction of Man Day on Project+0.0001))/Time to Get Exhaustion,
([(-0.5,0)-(2.5,2.5)],(-0.5,2.5),(-0.4,2.2),(-0.3,1.9),(-0.2,1.6),(-0.1,1.3),(0,1),(0.1,0.9),(0.2,0.8),(0.3,0.7),(0.4,0.6),(0.5,0.5),(0.6,0.4),(0.7,0.3), (0.8,0.2), (0.9,0),(1,0)))
Units: 1/Day
When AFMDP is the same as normal there is no increase in exhaustion level. Exhaustion rate is a function of 1-AFMDP, since denominator is constant. This value is also the measure of average “slack time”. Thus the exhaustion rate of work force is function of the compression in the average slack time. When 1-AFMDP

approaches zero and moves negative, people would not only be compressing their slack time but also working overtime.

Error Generation Rate=
Potential Productivity*Error Fraction*Effective WF
Units: tasks/Day
The rate of generating errors

Experienced Employees Turnover=
Experienced Employees Turnover Fraction*Experienced Workforce*Multiplier to ExpWorkforce Turnover due to Morale
Units: empl/Day

Increasing Work Rate=
(Workrate Sought-Actual Fraction of A Man Day on Project)/Work Adjustment Time
Units: 1/Day

Net Hiring=
Labor Resource Deficit/Hiring Adjustment Time
Units: empl/Day
Inflow to Workforce describing the hiring of new staff.

Newly Employees Turnover=
New Workforce*Newly Employees Turnover Fraction*Multiplier to NewWorkforce Turnover due to Morale
Units: empl/Day

Progress Rate=
Productivity*Effective WF
Units: tasks/Day
The rate of successful processing of tasks

Rate of Depletion of Exhaustion=
IF THEN ELSE (Exhaustion Increasing Rate<=0, Exhaustion Level/Time Spend on Depletion, 0)
Units: 1/Day
Once a period of overwork ends, either because of threshold or cease in schedule pressure the workforce returns to normal work rate. The rate is modeled as first order exponential delay, with a time equal to 4 weeks

Reduce Scope Through Redesign=
IF THEN ELSE (Scope Growth>=(Error Generation Rate*Original Work to do/Total Work to Do
+Progress Rate*Original Work to do/Total Work to Do), Scope Reduction Rate*Original Work to do, 0)
Units: tasks/Day

Rework Discovery=
(Undiscovered Active Rework+Undiscovered Passive Rework)/Time to Discover Rework
Units: tasks/Day
The rate of discovering rework

Scope Growth=
Rework Discovery*Ripple Effect Strength
Units: tasks/Day
The amount of additional work that is created due to discovery of rework.

Auxiliary:

Active Error Fraction in Task Perceived Done=
Undiscovered Active Rework/(Total Project Size*Fraction Perceived Done+0.001)
Units: Dmnl

Active Errors Fraction=
WITH LOOKUP (Fraction Actually Done,
([(0,0)-(1,1)],(0,1),(0.1,1),(0.2,1),(0.3,1),(0.4,0.95), (0.5,0.85),(0.6,0.5), (0.7,0.2),(0.8,0.075),(0.9,0),(1,0)))
Units: Dmnl
Active error fraction in "Error Generation Rate"

Boost in Work Rate Sought=
Handled Man Days/ (Total Workforce*(Overwork Duration Threshold+0.0001))
Units: Dmnl

Breakdown=
IF THEN ELSE (Overwork Duration Threshold=0, Time+TIME STEP, 0)
Units: Day

Congestion and Communication Difficulties=
Total Workforce*Total Workforce*Multiplier to difficulties due to Team Size
Units: Dmn
The average team member's drop in productivity below his nominal value

Desired WF=
Indicated WF*Willingness to Change WF+Total Workforce*(1-Willingness to Change WF)
Units: empl

Effective WF=
IF THEN ELSE (Actual Fraction of A Man Day on Project>1, Actual Fraction of A Man Day on Project*Total Workforce, Total Workforce)
Units: empl
The value of total workforce does not equal to the value of effective workforce. When employees are

overworking, that is *AFMDP* is more than 1, *Effective WF* is bigger than total workforce.

Effort Perceived Remaining=
Work Perceived Remaining/Perceived Productivity
Units: Day*empl

Error Fraction=
IF THEN ELSE(Actual Fraction of A Man Day on Project>1, Nominal Error Fraction *Multiplier to ErrGen due to WF Mix*Multiplier to ErrGen due to Schedule Pressure*Multiplier to ErrGen due to Fatigue*Multiplier to Active Error Regeneration Due to Error Density *Multiplier to Error Fraction due to Morale, Nominal Error Fraction*Multiplier to ErrGen due to WF Mix*Multiplier to ErrGen due to Schedule Pressure*Multiplier to Active Error Regeneration Due to Error Density*Multiplier to Error Fraction due to Morale)
Units: Dmnl

Expected Completion Delay=
Time Required-Time Remaining
Units: Day

Fraction Actually Done=
Work Done/Total Project Size
Units: Dmnl
Fraction of the total work to be done that is actually completed

Fraction of Experienced Workforce=
Experienced Workforce/Total Workforce
Units: Dmnl

Fraction Perceived Done=
(Undiscovered Rework+Work Done)/Total Project Size
Units: Dmnl
Fraction of the total work to be done that is perceived completed

Fraction Satisfactory=
1-Error Fraction
Units: Dmnl

Handled Man Days=
MIN (Max Shortage in Man Days to be Handle, Perceived Shortage in Man Days)
Units: empl*Day
When project is perceived to be behind schedule, the staffs need to handle the perceived shortage in man-days or the "maximum shortage in man-days" whichever is smaller.

Indicated Deadline=
Time+Time Required
Units: Day
Indicated completion date is used to adjust the projects formal “Schedule completion date”, where “indicated completion date” is the goal.

Indicated Productivity=
Productivity*Weight Given to Real Productivity + Potential Productivity*(1-Weight Given to Real Productivity)
Units: tasks/ (empl*Day)
If there were no perception delays this would be the best estimate of the productivity of the staff by taking into account the failure rate (detected faulty work) so far. See 'Perceived productivity'.

Indicated WF=
Effort Perceived Remaining/Time Remaining
Units: empl

KnownWork Remaining=
Total Work to Do
Units: tasks

Labor Resource Deficit=
Desired WF-Total Workforce
Units: empl

Max Shortage in Man Days to be Handle=
Max Boost in Man Hours*Overwork Duration Threshold*Total Workforce*Willingness to Overwork
Units: empl*Day
The maximum shortage employees can handle in a man-day

Morale=
IF THEN ELSE (Actual Fraction of A Man Day on Project>1, Nominal Morale*Muliplier to Morale due to Fatigue*Multipiler to Morale due to Schedule Pressure, Nominal Morale*Multipiler to Morale due to Schedule Pressure)
Units: Dmnl

Muliplier to Morale due to Fatigue=
WITH LOOKUP (Exhaustion Level,
([(0,0)-(100,10)],(0,1),(10,0.98),(20,0.93),(30,0.83),(40,0.79), (50,0.77),(100,0.77)))
Units: Dmnl

Multipiler to Morale due to Schedule Pressure=
WITH LOOKUP (Schedule Pressure,([(0,0)-(1,1)],(0,1),(0.2,0.95),(0.4,0.88),(0.6,0.76),(0.8,0.64), (1,0.5)))
Units: Dmnl

Multiplier to Active Error Regeneration Due to Error Density=
WITH LOOKUP (SMOOTH (Active Error Fraction in Task Perceived Done, 90),
([(0,0)-(1,10)],(0,1),(0.1,1.1),(0.2,1.2),(0.3,1.325),(0.4,1.45),(0.5,1.65),(0.6,1.95),(0.7,2.5),
(0.8,3.2),(0.9,4.1),(1,5.5)))
Units: Dmnl

Multiplier to ErrGen due to Fatigue=
WITH LOOKUP (Exhaustion Level,
([(0,1)-(100,2)],(0,1),(10,1.02),(20,1.07),(30,1.17),(40,1.21), (50,1.23),(100,1.23)))
Units: Dmnl
Multiplier to *Error Generation* due to Fatigue. It just affects the value of Error Fraction when working overtime

Multiplier to ErrGen due to Schedule Pressure=
WITH LOOKUP (Schedule Pressure,
([(-0.4,0.9)-(1,2)],(-0.4,0.9),(-0.2,0.94),(0,1),(0.2,1.05), (0.4,1.14),(0.6,1.26),(0.8,1.36),(1,1.5)))
Units: Dmnl

Multiplier to ErrGen due to WF Mix=
WITH LOOKUP (Fraction of Experienced Workforce,
([(0,1)-(1,2)],(0,2),(0.2,1.8),(0.4,1.6), (0.6,1.4), (0.8,1.2),(1,1)))
Units: Dmnl

Multiplier to Error Fraction due to Morale=
WITH LOOKUP (Morale,
([(0,0)-(1,2)],(0,1.3),(0.2,1.15),(0.4,1.09),(0.6,1.05),(0.8,1.02),(1,1)))
Units: Dmnl

Multiplier to ExpWorkforce Turnover due to Morale=
WITH LOOKUP (Morale,
([(0,0)-(1,10)],(0,3),(0.2,2.25),(0.4,1.7),(0.6,1.35),(0.8,1.125),(1,1)))
Units: Dmnl

Multiplier to NewWorkforce Turnover due to Morale=
WITH LOOKUP (Morale,
([(0,1)-(1,10)],(0,6),(0.2,3.5),(0.4,2.4),(0.6,1.7),(0.8,1.25),(1,1)))
Units: Dmnl

Multiplier to Overwork Due to Exhaustion=
WITH LOOKUP (Exhaustion Level/Max Tolerable Exhaustion,
([(0,0)-(2,1)],(0,1),(0.1,0.9),(0.2,0.8),(0.3,0.7),(0.4,0.6),(0.5,0.5),(0.6,0.4),(0.7,0.3),(0.8,0.2),
(0.9,0.1),(1,0),(1.1,0),(1.2,0),(1.3,0),(1.4,0),(1.5,0)))
Units: Dmnl

Multiplier to Productivity due to Morale=
WITH LOOKUP (Morale,([(0,0)-(1,1)],(0,0.3),(0.2,0.6),(0.4,0.74),(0.6,0.85),(0.8,0.95),(1,1)))
Units: Dmnl

Nominal Error Fraction=
WITH LOOKUP (Fraction Actually Done,
([(0,0.1)-(1,0.25)],(0,0.25),(0.2,0.24),(0.4,0.22),(0.6,0.17),(0.8,0.15),(1,0.145)))
Units: Dmnl

Overwork Duration Threshold=
Nominal Overwork Duration Threshold*Multiplier to Overwork Due to Exhaustion
Units: Day
This is a threshold beyond which employees would not be willing to work at an "above -normal" rate

Perceived Productivity=
SMOOTHI (Indicated Productivity, Average Time to Perceive Productivity, 0.04)
Units: tasks/ (empl*Day)
Computed from 'Indicated productivity' as a perception delay ('smooth')

Perceived Shortage in Man Days=
Labor Resource Deficit*Time Remaining
Units: empl*Day

Potential Productivity=
Fraction of Experienced Workforce*Nominal Potential Prod of Exp Employee
+ (1-Fraction of Experienced Workforce)*Nominal Potential Prod of New Employee
Units: tasks/ (empl*Day)

Productivity=
IF THEN ELSE (Actual Fraction of A Man Day on Project<=1, Actual Fraction of A Man Day on Project*(1-Congestion and Communication Difficulties) *Fraction Satisfactory*Potential Productivity*Multiplier to Productivity due to Morale, Fraction Satisfactory*Potential Productivity*(1-Congestion and Communication Difficulties) *Multiplier to Productivity due to Morale)
Units: tasks/ (Day*empl)
How many tasks are done by individual employee per day of 8 hours

Retirement Fraction=
WITH LOOKUP (Fraction Actually Done,
([(0,0)-(1,1)],(0,0),(0.1,0),(0.2,0),(0.3,0),(0.4,0.01),(0.5,0.02),(0.6,0.03),
(0.7, 0.04), (0.8, 0.1),(0.9,0.3),(1,1)))
Units: 1/Day
The fraction of how much undiscovered active errors retire to undiscovered passive errors at every time unit.

Schedule Pressure=
Perceived Shortage in Man Days/Effort Perceived Remaining
Units: Dmnl

Time Remaining=
Deadline-Time
Units: Day
Time remains in the project according to "Deadline"

Time Required=
KnownWork Remaining/Perceived Productivity/Desired WF
Units: Day
Time perceived to need to complete the project

Time to Discover Rework=
WITH LOOKUP (Fraction Perceived Done,
([(0,0)-(1,400)],(0,300),(0.1,300),(0.2,300),(0.3,300),(0.4,285),(0.5,240),(0.6,105),(0.7,60),
(0.8, 45), (0.9,30),(1,30)))
Units: Day
Table describing the relationship between project progress (fraction of work perceived completed) and the average time it takes to detect faults in work reported completed.

Total Work to Do=
Rework to do+Original Work to do
Units: tasks

Total Workforce=
Experienced Workforce+New Workforce
Units: empl

Undiscovered Rework=
Undiscovered Active Rework+Undiscovered Passive Rework
Units: tasks

Weight Given to Real Productivity=
WITH LOOKUP (Fraction Perceived Done,
([(0,0)-(1,1)],(0,0),(0.2,0.1),(0.4,0.25),(0.6,0.5),(0.8,0.9),(1,1)))
Units: Dmnl
Describes the transition from assessing productivity by pure intuition (project start) (when this weight is zero) to assessing productivity totally from observed data (when this weight becomes unity). In between this weight assumes a value between zero and unity.

Willingness to Change WF=
WITH LOOKUP (Time Remaining,

([(0,0)-(2000,1)],(0,0),(90,0),(180,0),(270,0.1),(360,0.3),(450,0.7),(540,0.9),
(630,1),(720,1),(810,1),(900,1),(990,1),(1080,1),(1170,1),(1260,1),(1350,1)))
Units: Dmnl

Willingness to Overwork=
IF THEN ELSE (Time>=Time of Last Breakdown+DeExhaust Time Control, 1, 0)
Units: Dmnl
The variable is set to 0 when maximum exhaustion level is reached and the overwork duration threshold is driven to zero

Work Perceived Remaining=
Total Project Size*(1-Fraction Perceived Done)
Units: tasks
How many tasks are perceived remaining at the present time.

Workrate Sought=
(1+Boost in Work Rate Sought)*Nominal Fraction of Man Day on Project
Units: Dmnl

Constant Value:

Average Time to Perceive Productivity=
180
Units: Day
Average time to perceive the current value of productivity. There are perception delays involving reporting, etc.

Average Assimilaton Delay=
80
Units: Day

Experienced Employees Turnover Fraction=
0.0001
Units: 1/Day

Hiring Adjustment Time=
90
Units: Day
The average time to close the gap between 'Desired workforce' and 'Workforce'.

Initial Deadline=
1200
Units: Day

Initial project size=
1200
Units: tasks

Max Boost in Man Hours=
1
Units: Dmnl
The maximum boost that people can increase their working rate. In this case it can be boosted by man 100%

Max Tolerable Exhaustion=
50
Units: Dmnl
The maximum tolerable level of exhaustion. At this point people will refuse to work overtime.

Multiplier to difficulties due to Team Size=
0.0001
Units: Dmnl

Newly Employees Turnover Fraction=
0.0002
Units: 1/Day

Nominal Fraction of Man Day on Project=
0.6
Units: Day

Nominal Morale=
1
Units: Dmnl

Nominal Overwork Duration Threshold=
50
Units: Day

Nominal Potential Prod of Exp Employee=
0.04
Units: tasks/(empl*Day)

Nominal Potential Prod of New Employee=
0.02
Units: tasks/(empl*Day)

Ripple Effect Strength=
0.3
Units: Dmnl
Multiplier reflecting the amount of additional work created and required when a work package is discovered to need a change.

Schedule Adjustment Time=
180
Units: Day

Scope Reduction Rate=
0.08
Units: Fraction/Day
The percent of original scope reduced each day through redesign

Time Spend on Depletion=
20
Units: Day
Time spend on depletion exhaustion

Time to Get Exhaustion=
1
Units: Day

Work Adjustment Time=
14
Units: Day
Time to adjust changing AFMDP